DELIS
Dynamically Evolving, Large-scale Information Systems

Integrated Project
Member of the FET Proactive Initiative Complex Systems

Deliverable D6.2.1

Intermediate Report on Concepts for Combining Locality with Hashing
Start date of the project: January 2004
Duration: 48 months
Project Coordinator: Prof. Dr. math. Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn, Germany
Due date of deliverable: December 2004
Actual submission date: January 2005
Dissemination level: PU – public

Work Package 6.2: Enhanced Distributed Hash Tables for Keyword Search
Participants: Heinz Nixdorf Institute, University of Paderborn, Germany
Max Planck Institute for Computer Science (MPII), Saarbrücken, Germany
Computer Science Institute (CTI), Patras, Greece
Authors of deliverable: Christian Schindelhauer (schindel@upb.de)
Peter Mahlmann (mahlmann@upb.de)
1 Introduction

In this intermediate report we present the state of the art for locality in distributed hash tables (DHTs) and discuss new approaches for improving locality features in DHTs. We begin with a brief introduction into distributed hash tables. Then, we define the three aspects of locality in this area: network locality, information locality, and interest locality. We discuss the nature of these locality measures and show that providing these locality measures in DHT based data structures is highly non-trivial. Then, we reconceive previous DHT based systems under these aspects. Eventually, we present new methods for improving locality in distributed hash tables.

1.1 Distributed Hash Tables (DHTs)

Consistent Hashing, aka. Distributed Hash Table (DHT), is introduced in [21] as a data structure for proxy-caching in the Internet to relieve so-called “hot spots”. It solves the problem to distribute data onto a set of servers\(^1\). As operational primitives we have insertion, deletion and lookup for the data elements, as well as insertion and deletion of servers. The following objectives play the main role:

- Efficiency according to time behavior and memory consumption.
- Load balancing, i.e. the data size on the servers should be fairly balanced.
- Consistency: If a server is inserted, then only those elements need to be rearranged which are placed on the new server. Analogously if a server leaves the network then only data from this server needs to be reassigned to other servers.

The solution of Karger et al. [21] uses an elegant application of standard hash functions. Servers and elements are mapped onto intervals according to a hash function which “swirls” them to positions as if they were randomly inserted into the intervals. Then, an element is assigned to the nearest server of this interval, see Fig. 1. This simple, elegant scheme provides both load balancing and consistency, and plays an important role in further developments of modern distributed storage systems, like storage area networks (SAN) and peer-to-peer-networks. Note that in the original work this DHT is enhanced by a technique, called views, which allows locality. We will come back to this feature later in this report.

For proxy caching in the Internet, as well as for storage area networks, the number of servers is relatively small which is not the case for peer-to-peer-networks. Here, the number of servers, called peers, is so high that a peer cannot know all participants in the network. So, peers see only a small fraction of the network. For describing the set of nodes known to a specific node one uses directed graphs, such that a directed edge from peer \(u\) to peer \(v\) indicates \(u\) knowing \(v\) (and not necessarily vice versa). This reduces the description of peer-to-peer-networks to the task of finding appropriate graph structures that provide the following features.

1. Scalability: The in-degree and out-degree of each node is small. The graph structure is flexible enough to insert and delete nodes causing only a small (local) impact to the graph structure.
2. Efficiency: The graph structure allows to efficiently lookup the peer responsible for an element by a small number of hops.

The first approach providing (some) scalability and efficiency is CAN (A scalable content-addressable network) [27]. It combines a DHT with a \(k\)-dimensional hash space and a lattice network, where

\(^{1}\)Here, we use the terms server, node, peer, and host as the data storage device and the terms item, element, file and data as the data to be stored on the storage devices synonymously.
peers are assigned to rectangular regions and connected to their neighbors in all $2k$ directions, see Fig. 2. This way the maximum lookup time is a $k$-th root of the number of peers while inserting and deleting peers in this network can be easily implemented.

This approach is improved by the Chord-network [35] which uses a ring as a hash space for peers and elements and provides $O(\log n)$ edges as shortcuts for the lookup of data elements on the ring. So, the lookup time reduces to $O(\log n)$, see Fig. 3.

Most efficient peer-to-peer-networks are based on distributed hash tables. Peers as well as data elements are mapped to a hash space. The positions of the peers in this space define regions for each peer. All data elements mapped inside a region assigned to a peer are stored at the corresponding data element. These networks benefit from the main property of such a DHT: after inserting a new peer only a small fraction of data elements needs to be reassigned (and vice versa when a peer is leaving), while the balance of data can be maintained. These DHTs are used in peer-to-peer networks CAN, Chord, Pastry, Tapestry, SkipNet and many more [12, 16, 20, 23, 24, 25, 27, 28, 35].

Recent research is mainly concerned in improving the load balance between the peers and optimizing network parameters like small diameter, small degree, low-latency using probabilistic or deterministic algorithms. Within this report we concentrate on locality in such networks.

### 1.2 Locality

One of the goals of the work-package 6.2, Enhanced Distributed Hash Tables for Keyword Search, within the DELIS project is to provide data structures for distributed web search engines. For this, locality plays an important role. We will see that providing locality features for DHTs as used in peer-to-peer networks is the most challenging task. Solutions capable to solve this case can be easily generalized for storage area networks and web caching. From now on we focus on peer-to-peer networks and consider the following aspects of locality.

#### 1. Network Locality

A peer-to-peer-network provides network locality if lookup operations can be performed with small latency.

This notion refers only to peers of the network. Often in peer-to-peer networking designers abstract from the underlying network (the Internet). Networks are constructed to provide small hop distance. Yet, a hop connecting computers in Greece and Australia counts as much a hop from one room to the next one of a building. Clearly, the locality of peers within the

Figure 1: The working principle of consistent hashing, aka. distributed hash tables.
Internet has to be taken into account for optimizing the network structure and the data lookup operations.

2. Information Locality

A peer-to-peer-network provides information locality if closely related data elements are stored on network-wise close peers.

Distributed hash tables map data elements to (pseudo) random positions via the hash functions. Elements stored at a peer are completely unrelated. So, a structured query like a range query or a proximity search cannot be performed efficiently using a plain distributed hash table.

3. Interest Locality

A peer-to-peer-network provides interest locality, if peers can choose on providing lookup service and data storage for certain data. If peers choose to provide certain data, then the network structure allows efficient lookup to data relevant to a peer.

In the Web certain data is intrinsically local, e.g. most of all Polish web-sites are created in Poland and accessed from computers in Poland. Hence, it makes a lot of sense to store such data on Polish peers. Another aspect is that in peer-to-peer networks often data is published and distributed which inflict with local law, e.g. violating copyright or containing offending contents. In a peer-to-peer network participants should be able to reject storing such data or even reject the assistance for looking up such data.

Why locality is a Problem in DHT

Usually, most peer-to-peer-networks do not provide locality. The reason is the intrinsic non-locality of the distributed hash table. We exemplify this for the Chord-network [35].

In a Chord-network every peer is given a random position on the ring. So, the ring neighbors as well as the \(O(\log n)\) other neighbors indexed by so-called fingers are randomly chosen from the whole
Figure 3: A Chord peer-to-peer-network. For visibility only for node 0 the fingers are depicted as dashed arrows

set of peers. Therefore, the latency of nearly all edges equals the average latency time of two peers in the network. This clearly violates the network locality property.

For the information locality note that every element is placed at a (pseudo) random peer according to the hash function. Related data elements are torn apart and stored on different places of the network. So, information locality is not given at all.

Similarly, interest locality is not provided by Chord. Peers have to store every information assigned by the consistent hash function. If peers reject information entries, then the information is not stored at all. Furthermore, peers cannot apply for storing indices they are interested in.

This example shows how locality can be completely prevented by the use of the distributed hash table in the chord network. We now give deeper discussion of the three aspects of locality.

1.2.1 Network based locality

As we have stated above network locality is provided if lookup operations can be performed with small latency. Peer-to-peer-networks as overlay network often regard each hop of the network equally. However, latencies in the Internet can differ a lot. Traffic patterns are bursty [13] causing continuously varying packet latencies. Routing tables of the Internet adjust several times each day. So, the routes in the Internet are unpredictable, since the routing tables are built by distributed mechanisms. Furthermore, the interconnections between routers are not totally known and are part of serious research [14].

Nevertheless, from all three types of locality, this aspect is the one where most of the research has been done so far. The standard measure for locality is the latency of packets between peers. A standard assumption is that this measure obeys the triangle inequality (being a metric) or can be approximated by a Euclidean metric. Both assumptions do not necessarily reflect the situation in the Internet.

The bursty behavior of the Internet traffic causes quickly changing traffic patterns in the Internet.
So, measuring the (average) latency needs a considerable amount of ongoing measurement. This problem increases if the peers are connected with a low speed connection to the Internet. Then, the fast Internet backbone connection only plays a minor role in the packet latency, whereas the individual traffic blocking the low-speed dial-up connection disturbs estimation of the average latency.

Recent research shows that the autonomous systems as the nodes of the Internet obey a power law [33]: The number of connections between these autonomous systems are distributed according to a power law (Pareto distribution) of degree of about 2. One can conclude from this, that the diameter of the routers of the Internet is smaller than one would observe in an Euclidean metric. In fact, the lower time bound induced by the speed of light for transmitting information sending information around the globe is often much smaller than the overhead of congested undersized routers in local area networks.

Knowing the difficulties using the latency based network locality model we will present an approach in subsection 3.1 which is based on the routing tables of the Internet which provides longer lasting and more reliable information about the structure of the Internet.

1.2.2 Information locality

Before we discuss how information locality can be provided for distributed hash tables we shortly discuss what it means that elements are closely related. One can distinguish between syntactic and semantic relation of information.

The easiest way to relate data syntactically is their lexicographical ordering. Words are arranged in a one-dimensional array and words with long common prefixes are closely neighbored. Another more practical relationship is given by the word distances, where words are close to words if they are a sub-string or if they differ only in a small number of letters. This metric can be used for detecting typographical errors. Other approaches define a similarities of words by phonetic distances [36] or word stemming. This case is much more difficult to handle compared to the lexicographical ordering, since this metric cannot be embedded into a Euclidean space. Besides this approach numerous other notions of syntactical neighborhood are conceivable, e.g. involving character encodings, prefixes. In this context we see numbers or positions (e.g. points in a map) as being rather syntactical than semantical (from a behavioral point of view).

For semantical relationship the situation is even more complicated. Even for a human expert it is hard to tell how close the relationship between two given words is. The only chance of algorithms getting some grip on this problem is to provide them with a large database or some dedicated structured information. Another solution is to utilize users for making these decisions if necessary. Questions like these are part of a related work package in DELIS 6.1, Models and Strategies for Collaborative Web Information Service, where we refer to here.

For the following work we assume that given two pairs of words we have a black box algorithm which can decide which one of them is closer to each other and that this decision is consistent to an underlying metric (i.e. obeying the triangle inequality). Since the structure of locality is highly depending on this black box operation it is difficult to optimize the network structure. On the bottom line it seems that at least for the semantic information neighborhood DHTs providing information locality can only be experimentally tested and compared to other approaches.

1.2.3 Interest locality

Recall that a peer-to-peer-network provides interest locality, if peers can choose on providing lookup service and data storage for such data. If peers choose to provide certain data, then the network structure allows efficient lookup to data relevant to a peer, see also [10] for a more detailed discussion.

- Local Caching (egoistic caching)
When data are downloaded, then the downloading peer can store (i.e., cache) the data and thus serve as a server. On the other hand, a peer can reject storing data in which it is not interested. Such egoistic caching and storage systems are usually not covered by efficient data structures named above. As an example, see the currently very popular Bittorrent system that utilizes the upstream of file down loaders for improving the overall transmission capacity. However, Bittorrent cannot be seen as a data storage system because the lookup of data is not provided by the system. It only provides an efficient method for distribution of one dedicated file.

When data are downloaded, then the downloading peer can cache the data and thus serve as a server for his neighborhood. However, if a peer has no interest in downloading data, then it does not store data for other peers. Such a scheme is called egoistic caching. Recently we have developed and implemented such ideas combined with a lookup service in a peer-to-peer network based on a mobile ad hoc network.

- Local Storage

In the Web, certain data is localized to countries, regions, subgroups, communities, companies etc. Storing data elsewhere does not make sense or is even not desired. Common practice in peer-to-peer networks is to store at least links pointing to some information. Of course, one could encrypt these entries, as in Freenet of the Free Haven Project. However, this violates common practice in the Internet where information stored on computers is kept transparent. From a social point of view, the acceptance of such a peer-to-peer system would be rather low if such information hiding takes place. Especially, if a user can come into conflict with law because his/her computer is unknowingly utilized for criminal actions.

Interestingly, this problem of interest locality is already addressed by the first paper on consistent hashing. However, there a centralized approach is presented that is not applicable to distributed computer data storage systems like peer-to-peer networks, see subsection 2.2.

2 State of the Art

In this section we focus on some important research concerning distributed hash tables and locality. We have chosen these examples because of their importance to the area of distributed hash tables even if only little locality is provided.

2.1 First and second generation peer-to-peer networks

The general interest in peer-to-peer networks started with the hype of the Napster network, which actually provided a client-server data structure for a centralized index of files distributed in the Internet. Because of the dedicated server this network can only be seen as a predecessor of real peer-to-peer networks.

We regard Gnutella as the first peer-to-peer network. It treats all participating hosts equally. However, it fails to provide efficient lookup algorithms. Yet, by storing information only on the peers providing and querying files, it provides perfect interest locality. On the other hand, neither network locality nor information locality are provided (if one neglects the information locality induced by the interest locality).

This problem is solved in the second generation peer-to-peer networks Gnutella (current version) and Kazaa, where some dedicated super-peers are used for indexing while data is stored distributedly on the peers. These super-peers are responsible for lookups which are performed (as in Gnutella) by a query broadcast among the super-peers. The choice of the super-peers in these so-called hybrid networks provides some network locality. Besides the index entries stored on the super-peers interest...
locality is provided, too. Again these peer-to-peer-networks do not provide information locality. Furthermore, efficiency and scalability are also not given, since a lookup operation affects all super-peers.

All these early approaches suffer from bad performance (for a large number of lookups). This is caused by the fact that they do not use any efficient network structuring like distributed hash tables.

2.2 Web caching by consistent hashing

As we have already mentioned distributed hash tables are introduced in [21, 22]. Notably, they provided a mechanism for locality called views. A view is a subset of servers covering a region, thus enabling network locality. Elements are stored on each view on the hosts which is nearest to the hashed value of the element. This way, the number of copies is kept small while every element is available in all views. These views are not necessarily disjoint and can also serve for providing some coarse-grained information locality although not being introduced for this reason. However in peer-to-peer-networks such a locality mechanism cannot be applied, since the views are manually administered according to the placement of the servers in the Internet.

Interest locality is clearly violated because of the copies of elements in all views. Besides this it is not clear how information locality can be provided within one view.

2.3 Storage Area Networks

Another application area for distributed hash tables are storage area networks. Here, the servers are hard disks which resemble to a large virtual disk [29, 4]. The distributed hash tables define the index table mapping the hard disk memory units to the virtual disk. Such storage area networks benefit from parallel disk reads and writes and provide small latencies.

This is the reason why any form of locality as described above is not desirable. On the contrary, if data elements are needed at the same time because of interest locality, then in the best case these parts are distributed among all available hard disks providing maximum parallelism. The same observation holds for information locality. Network locality does not play any role, since the decisive parameter for time are the access rates to the hard disks and not the interconnection between the storage devices.

2.4 CAN: Content Addressable Network

The first efficient approach to peer-to-peer networks using distributed hash tables is CAN [27]. The efficiency is asymptotically rather mediocre providing a hop distance of $O(n^{1/c})$ for some constant natural number $c$ compared with later published peer-to-peer networks.

Interestingly, the authors cover two aspects of locality (unlike many other peer-to-peer-networks solely concentrating on network locality). Only interest locality was not covered by this landmark paper. For the information locality the authors suggest the following variant. They map the data elements directly onto the Euclidean space according the indexes without using a hash function. For the load balancing the peers are placed top-down in the tree structure of the CAN-network. This approach gives very good information locality, if the information can be mapped to some $k$-dimensional space.

For the network locality two techniques are discussed. Zones (where all elements are mapped to the same peer) can be overloaded. Then all elements of a particular zone are stored on all peers belonging to that zone. A peer in a zone knows all peers in neighbored zones and the relative distance

\footnote{According to this observation considering these networks is clearly out of the scope of this intermediate report. We have included this discussion for covering all types of peer-to-peer networks.}
measured by the round trip time. For the lookup each peer contacts the neighbor which is the closest one providing a fast lookup procedure.

Another approach uses \( m \) landmarks placed in the network. Each peer measures the round trip time to these landmarks. The landmarks are sorted and each of the \( m! \) orderings determines a special area of the hash space. Now the \( m! \) areas are placed such that neighbored zones can be reached by flipping the ordering of two closely landmarks. So, routing through the CAN overlay network should use small latency routes in the Internet.

### 2.5 Pastry and Tapestry

Pastry [28] and Tapestry [16] rely on the efficient routing of Plaxton et al. [26]. For this, each peer is assigned an address consisting of a string over an alphabet \( \Sigma \) (typically of cardinality 16). Now the routing table of each peer has an entry for each prefix \( p \) and followed by a symbol \( s \) of the alphabet. From all peers with prefix \( ps \) the closest one is added to the routing table (e.g. measured by the RTT). This way, Pastry and Tapestry provide very good network locality which can be experienced in implementations available for both systems.

The problem of interest and information locality is not addressed by these peer-to-peer-networks.

### 2.6 CHORD and DHash++

Plain Chord [35] does not provide any locality. Dabek et al. [9] overcome this problem by using a variety of techniques. The main method is adapting the link structure to the message latencies using proximity neighbor selection. Note that the \( O(\log n) \) fingers of Chord providing shortcuts through the ring can be changed by a constant factor without changing the asymptotic hop length for lookup. So, the authors choose the best position for each finger (or approximate this best position by sampling at some positions of the ring). Another interesting technique is the usage of erasure-resilient codes of [1] providing smaller latencies than data replication. Further techniques, involve the use of sequential lookup operations, integration of routing and fetching, and a specially designed transport protocol. The bottom line is that using these techniques the network locality of Chord can be improved up to the level of Pastry and Tapestry.

However, interest and information locality still remain unsolved for Chord networks.

### 2.7 Skipnet

Skipnet [25] is a recent interesting approach to adopt information locality into peer-to-peer-networks. The elements are arranged on a ring and peers are uniformly distributed over this set and not over a hash space. Then, every peer tosses a series of coins giving a (long enough) sequence of random bits. For all prefixes \( p \), all nodes with the same prefix \( p \) are connected ring-wise like the original ring omitting all peers not sharing the prefix \( p \). This way one achieves a data structure similar to skip-lists of height \( O(\log n) \) providing information locality.

Skipnet provides an efficient peer-to-peer network structure with some information locality and has the potential for also providing interest locality. It is not clear whether network locality can be achieved in this network structure as well.

### 3 New Results and Techniques

In work package 6.2 we investigate the following methods for peer-to-peer networks for providing network, information and interest locality.

- Network locality detection and control
• Self-organizing random networks
• Top-down distributed hash tables using weighted consistent hashing
• High-degree peer-to-peer networks

We now give a more detailed description.

3.1 Network locality detection and control

The common approach for network locality is based on a metric described by the round trip time (RTT) of messages between hosts. This latency matrix is used to minimize latencies in the peer-to-peer network.

There are several drawbacks of this approach:

• The variance of the RTT is often very high. Therefore the RTT needs to be averaged over long time periods using continuous measurement.
• The routing in the Internet changes from time to time. Then, averaging over long time periods gives a wrong delayed image of the latencies.

Our solution to this problem is a clever use of the traceroute-routing provided by the Internet Control Message Protocol (ICMP). The feedback of a traceroute is not only the RTT between two nodes, but also the path and the RTT to all interior routers of the path. Using this data it is possible to draw an accurate timely picture of the inter-networking between the participating peers. However it does not make sense to invoke this routine for all peers participating in the network. Yet, if one reduces the number of such calls, then only an approximate picture of the Internet can be detected.

Currently, “Peer-Near” is being developed as a master’s thesis by a diploma student [30]. When finished in February 2005 it provides a tool which reflects the accurate routing and latencies between the participating nodes. As a first approach this tool works in a client-server structure. It is planned to be extended to a fully peer-to-peer network approach, as being done in DimesNet [14] for a different purpose.

3.2 Top-Down Networks

From our point of view peer-to-peer networks are usually built in a bottom-up fashion. The underlying assumptions are the following.

• The first contact is a (latency-wise) near node
• It is easier to guarantee connectivity in a small-degree basic network like a ring (Chord), a grid (CAN), a butterfly-graph (Viceroy), etc.

We do not think that these assumption hold in general.

First, in many implementations of peer-to-peer-networks the code is downloaded from some central servers providing a list of standard peers. Clearly such an initialization does not provide much locality. After the first run, some network neighbors are stored locally at each peer. Now, fluctuation of network participants is very high. So, many links need to be tested for finding near neighbors of former sessions. Therefore, locality is not given for free, yet must be achieved by appropriate methods.

Using such a map of the Internet routing one can use techniques of CAN, DHash++, Pastry, or Tapestry to reduce the latency of the routing. We will use this information to provide local entries in the data structure. For this we use a layered distributed hash table, where in each layer only nodes
of regional networks reside. This information can be detected by the routing structure as well as by the latency matrix delivered by Peer-Near.

The lowest layer of the Distributed Hash Table describes the LAN of a node. Then, there will be a layer describing local neighborhood and regions. In the upper-most layer all nodes of a peer-to-peer network are connected. Thus, the peer-to-peer network can be seen as a union of (possibly intersecting) local regions, which recursively provide the same structure, see Fig. 4.

### 3.3 High-degree Networks

There has been a lot of effort in the algorithmic community to provide basic networks of very low degree, i.e. with a small number of connections for each link. This small degree imposes a vulnerability to the networks, exploited by attackers who can easily isolate peers, or by the regularly disconnecting peers. The odds of being disconnected from the network exponentially decrease in the degree of the each node. So, it seems absurd to try to improve on this parameter, to save a minimum amount of memory, while each node is supposed to store large amounts of data elements.

Furthermore higher degree can decrease the diameter and the routing latency of a network. In many peer-to-peer networks load balancing is done using the principle of multiple choice (see Viceroy or Distance Halving). If one uses this technique, then the basic small degree network does not reflect any locality. As a consequence, in networks like Chord the last hop is the most expensive hop (regarding latency). It is as expensive as the average distance between two nodes of the network. So, even the look-up of a peer to a near neighbor storing the information uses nodes of the network located far away.

The bottom line is that it makes sense to consider a robust high degree connection network that provides connectivity on the upper-most level. However, little is known about the maintenance of large degree networks. The main problem is to guarantee connectivity without using a designated small network structure like a ring.

### 3.4 Random Graphs as Nodes of the Top-Down-Network

Our solution is a random graph with at least logarithmic degree. For this, we have developed a simple strategy that checks and renews old and broken links of such a random network and calculates the latency distance between neighbors [31]. We can provide connectivity if no nodes leaf without notice.
If a constant number of nodes leave without notice then this network remains connected with high probability, i.e. $1 - n^{-O(1)}$ where $n$ is the number of nodes.

Furthermore, there is a simple update rule that establishes an expansion property from every starting state (as long as the starting state is connected). As soon as the expansion property is established it is maintained with high probability. Such expander graphs provide small diameters. As a consequence, this construction gives for large (polynomial) degree of $n^{1/k}$ with some constant $k > 1$ a constant diameter network.

Using this construction one can for the first time construct a peer-to-peer-network top-down. On the upper-most layer one has a random graph which already provides small diameter and connectivity. Then, this network is subdivided in some $\Theta(g)$ sub-networks, where $g$ denotes the degree of the random network. For the next layer this construction is repeated until the sub-networks are cliques. For polynomial degree this network allows an expect constant hop routing algorithm.

Note that this top-down structure allows to exploit locality structures and to assign peers to sub-networks of the same region. Unlike as in (all the other) bottom-up networks it is now possible to remain in a region if the request and its answer reside in the same region.

### 3.5 Self-Organized Random Networks

The operation for constructing the random backbone can also be used to express interest and information locality. For this the strategy of 3.4 is only applied if the locality increases. Such a network will be an additional overlay network providing shortcuts and first guesses before starting the standard routing of the efficient top-down peer-to-peer network. This approach follows the research results of Babaoglu et al. [17, 18] who showed empirical evidence for the self-organizing behavior of such approaches. In this context we use such a network as a heuristic extension and not as a replacement of an otherwise controlled network.

### 3.6 Weighted Consistent Hashing

Peer-to-Peer networks connect heterogeneous peers. In practice most of the peers provide small storage, small transmission bandwidth. Most of the peers are requesting more files than they are providing. On the other side there are some providers with several magnitudes higher storage and bandwidth. This states the problem of distributing data in the peer-to-peer network. There are other approaches providing a scheme that distinguishes good peers for storing data from bad peers requesting more files than they are downloading, see Kazaa, Gnutella II, and others [15]. This knowledge is used to build up the communication network structure. Detecting such peers is part of the WP 6.5. of DELIS, Incentives for Collaborative Behavior and Fairness Metrics.

We take a different approach. We store on each peer a portion of the data which is proportional to the available storage or bandwidth (usually high storage peers also come with high bandwidth connections). Such a data balancing is called fair. For homogeneous nodes new techniques for dynamic load balancing are presented in [2]. We now discuss the heterogeneous case.

The straight-forward approach to solve this weighted balancing is to introduce virtual copies of a peer and thus increase the share proportionally to the number of nodes. Since in practice this would mean introducing more than thousands of copies. Brinkmann, Scheideler, Salzwedel et al. [4] provide a more efficient approach for a SAN called Presto. However they do not achieve a completely fair balance. Small weighted peers receive too few data, yet the share among peers with large weight is distributed in a fair fashion. Another drawback is that this weighted consistent hashing (designed for storage area networks) cannot be used for peer-to-peer-networks, since storing data elements requires full knowledge of the weights of all peers.

We provide in [32] a simple scheme that achieves perfect balance. It is very similar to the standard approach of distributed hash tables and uses for each peer only a logarithmic number of copies. The
main advantage of such a weighted consistent hash-table is that hashing can be done recursively in a tree structure reflecting the different capabilities of the branches. This means that in a peer-to-peer network data peers can be organized by regions. Then, the data in this uppermost sets can be distributed fairly among all peers. This approach can be recursively repeated to the lowest layer.

This technique is a building block for the top-down peer-to-peer network approach allowing to provide network and interest locality.

4 Result Dissemination

The research in this WP has led to the following publications in international conferences [2, 3, 32]. These publications are available on request from the coordinating site UPB. Other publications [30, 31] are in preparation and will be published by the beginning of 2005.

Some results described in this document have also been presented at the DELIS SP6 meeting in Oslo, July 26-27th. Furthermore, results and concepts have been exchanged on a research meeting of scientists of the University of Paderborn and the Max-Planck-Institute of Computer Science (MPII), Saarbrücken at the site of MPII on November 25-26th.

References


