



**PEPITO**  
IST-2001-33234  
**PEer-to-Peer Implementation and TheOry**

**Deliverable no: D2.8**

## **Second Progress Report on Distributed Algorithms and Services**

REPORT VERSION: first

REPORT PREPARATION DATE: 23rd January 2004

CLASSIFICATION: Public

DELIVERABLE NO: D2.8      DUE DATE: Month 24      DELIVERY DATE: Month 24

PROJECT START DATE: 2002.01.01      PROJECT DURATION: 36 months

RESPONSIBLE PARTNER: KTH

PARTICIPATING PARTNERS: KTH, SICS, EPFL, UCL

PROJECT COORDINATOR: Swedish Institute of Computer Science AB

PROJECT PARTNERS: EPFL Lausanne, INRIA Paris, KTH Stockholm, UCL Louvain, University of Cambridge UK

**Project funded by the European Community under the ‘In-formation Society Technologies’ Programme (1998–2002)**

Project Number: IST-2001-33234

Project Acronym: PEPITO

Title: PEer-to-Peer Implementation and TheOry

Deliverable No: D2.8

Second Progress Report on Distributed Algorithms and Services

Due date: project month 24

Delivery date: 2003.12.31

Responsible Partner: KTH

Participating Partners: KTH, SICS, EPFL, UCL

23rd January 2004

Prepared by Luc Onana Alima, with input from: Ali Ghodsi, Sameh El-Ansary, Seif Haridi, Per Brand, Rachid Guerraoui, Uwe Nestmann, Johannes Borgström, Bruno Carton, Valentin Mesaros, Peter Van Roy and Peter Sewell.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Objectives . . . . .	3
1.2	Achievements during the year 2003 . . . . .	3
<b>2</b>	<b>A framework for DHT-based overlay networks</b>	<b>4</b>
<b>3</b>	<b>Enhancements of the DKS overlay networks</b>	<b>5</b>
3.1	Network structure . . . . .	6
3.2	Stack embedded messages for robust recursive lookup . . . . .	6
3.3	A novel algorithm for the join operation . . . . .	7
3.4	A novel algorithm for the leave operation . . . . .	7
3.5	Novel algorithms for handling failures . . . . .	7
3.5.1	Failure semantics . . . . .	8
3.5.2	The novel approach for handling failures . . . . .	8
3.6	One-to-many communication primitives in DKS . . . . .	8
3.6.1	Broadcast . . . . .	8
3.6.2	Multicast . . . . .	9
3.7	Reducing communication with eager update . . . . .	9
<b>4</b>	<b>Simulation environment for peer-to-peer algorithms</b>	<b>9</b>
<b>5</b>	<b>Simulation-based comparison of Chord versus DKS</b>	<b>10</b>
<b>6</b>	<b>Formal specification and verification of DKS's algorithms</b>	<b>10</b>
<b>7</b>	<b>Symmetric overlay networks and high scalability</b>	<b>11</b>
<b>8</b>	<b>Probabilistic broadcast algorithms</b>	<b>11</b>
<b>9</b>	<b>Proof of concepts</b>	<b>12</b>
9.1	Implementation of peer-to-peer infrastructures . . . . .	12
9.2	Example applications . . . . .	12
9.2.1	DKS-based WebLog application . . . . .	12
9.2.2	DKS-Based file system . . . . .	12
<b>10</b>	<b>Cooperation with other workpackages in PEPITO</b>	<b>12</b>
10.1	Cooperation with WP 1 . . . . .	12
10.2	Cooperation with WP 3 . . . . .	12
<b>11</b>	<b>Information dissemination</b>	<b>13</b>
11.1	Published papers . . . . .	13
11.2	Papers in preparation for publication . . . . .	13
<b>12</b>	<b>Attached papers</b>	<b>14</b>

# 1 Introduction

In this document, we present the progress made in workpackage 2 (WP 2), devoted to distributed algorithms and services in the project PEPITO. The period covered by the report is from 2003.01.01 to 2003.12.31.

We proceed as follows. In Section 1.1, we recall the objectives of workpackage 2. In Section 1.2, we give a list of the major achievements during the year 2003. Then, from Section 2 to Section 8, we elaborate on each of the achievements listed. In Section 9, we summarize some of the works being performed to proof the applicability of our designs. Section 10 highlights the cooperation between workpackage 2 and some other workpackages of the PEPITO project. In Section 11, we show the results of our efforts in disseminating information. Finally, a list of attached papers is given in Section 12.

## 1.1 Objectives

As it became apparent in the first progress report, the main objective of WP2 is to provide a robust, low communication and scalable application-independent infrastructure that eases the construction of peer-to-peer applications. This entails design of a substantial number of distributed algorithms capable of supporting such an infrastructure. These include at least:

- ▷ Distributed algorithms for constructing overlay networks with desirable properties such as scalability, fault-tolerance and providing location-independent one-to-one and one-to-many communication primitives.
- ▷ Distributed algorithms for consistency and approximate replication.
- ▷ Distributed algorithms for a large-scale directory services.
- ▷ Probabilistic distributed algorithms that might prove powerful in environments with high dynamism.

## 1.2 Achievements during the year 2003

Substantial contributions have been made in WP 2 during the above mentioned period. These contributions are organized along the following items:

1. A framework for DHT-based overlay networks.
2. Enhancements of the DKS overlay networks.
3. A simulation environment for peer-to-peer algorithms.
4. A simulation-based comparison of the Chord system versus the DKS.
5. A formal specification and verification of DKS's algorithms.
6. Symmetric overlay networks and high scalability.
7. Probabilistic broadcast algorithms.
8. Proof of concepts
  - ▷ Integration of DKS's algorithms in the DSS.
  - ▷ A DKS-based WebLog application.

- ▷ A DKS-based file system.
- ▷ A peer-to-peer library within the Mozart/Oz platform.

In the rest of this document, we shall elaborate on each of these points in turn. Though the document is intended to be less technical, we opted to provide some technical aspects in order to help the reader appreciate the contributions. We shall, for most of the items mentioned above, give the motivations of the problem addressed and a brief description of the proposed solution.

## 2 A framework for DHT-based overlay networks

One of the main contributions of WP 2 has been the provision of a framework for designing structured overlay networks based on Distributed Hash Tables (DHTs).

In the first progress report on distributed algorithms and services, Deliverable No. 2.7., we summarized our framework for building efficient structured overlay networks. In the current report, we give a more elaborated version of this framework.

DHT-based overlay networks were introduced to construct overlay networks with a number desirable properties such as scalability, self-organization and high guarantees for locating items stored in the system.

To achieve scalability, all the DHT-based overlay networks we know of proceed as follows.

1. A large space with some notion of “distance” is chosen, we call it *virtual space* or *identifier space*.
2. A hashing function  $H$ , typically SHA 1, is used to map
  - ▷ each item  $d$  onto the virtual space. This assigns a virtual identifier to  $d$ , denoted  $H(d)$ .
  - ▷ each participating node  $n$  onto the virtual space. This gives  $n$  a virtual identifier, denoted  $H(n)$ .
3. A *structuring strategy* is exploited to construct a network that *spans* the whole virtual space. This network is called *overlay network*. Along with this network, a routing mechanism is provided.
4. Using the “distance” defined on the virtual space, each item  $d$  is assigned to the node  $n$  such that  $H(d)$  is “closest” to  $H(n)$ . We call such a node the *manager* of  $d$ .

The main challenge lies on the construction and the maintenance of the overlay networks. In the first progress report of WP 2, we summarized the framework that we have been building to address this challenge from a fundamental stand-point, instead of looking at it from the perspective of a specific system.

The proposed framework assumes virtual space of size  $N = k^L$ ,  $k \geq 2$ . That is, the virtual space size is an *exponential function of base  $k$* , where  $k$  is a system-wide parameter. Then, the framework provides four design principles. These are:

1. *Distributed  $k$ -ary search*: As a structuring strategy, we proposed the distributed  $k$ -ary search principle, in which the virtual space is systematically divided by  $k$ , and each node is assigned a routing table such as to ensure that the *diameter* of the constructed network is  $\log_k(N)$ . An entry of a routing table is a pointer to a node that is considered *responsible* for some portion of the virtual space. The logical division of the virtual space required is subject to the following constraint: for each node, the size of the routing table is either in  $O(1)$  or in  $O(\log_k(N))$ .
2. *Local atomic action*: To account for the high dynamism assumed in peer-to-peer systems, the framework proposes the use of local atomic action for inserting new nodes and for managing the departure of nodes that wish to cooperate while leaving the system. The local atomic action is important to ensure strong guarantees for locating any item inserted into the system.

3. *Correction-on-use*: As a technique for maintaining overlay networks, the framework proposes correction-on-use. This is a powerful technique that ensures that the bandwidth is going to be consumed only when needed. Interestingly, the technique is applicable to all *interval routing-based networks*<sup>1</sup>, which is another important characterization of all the overlay networks of interest in this document.
4.  $f + 1$  *replication*: To ensure availability, replication is used as in traditional systems. The proposed framework ensures that each item inserted into the system is kept at  $f + 1$  nodes.

All the DHT-based overlay networks we are aware of can be derived from this framework. And, as stated in Deliverable No. 2.7, to use our framework, the designer mainly has to focus on:

1. The *division* of the virtual space. The division of the space is guided by the routing mechanism to be used and can be either *fixed* or *relative*.
  - ▷ *Fixed division of the space*. In the fixed division of the space, there is a *virtual global* partitioning that is made for all potential nodes of the system. A node that joins the system will obtain a routing table by picking some nodes from some of the portions of the global partition. This is the case for systems such as Pastry[16], Tapestry[18], P-Grid[2] and Kademlia[12].
  - ▷ *Relative division of the space*. In the relative division of the space, for each node, there is a logical division of the virtual space that is specific to the node. Furthermore, any node  $n$  by knowing the identifier of any other node  $n'$ , can compute the logical division of the space relative to node  $n'$ . This is the case for systems such as Chord[17], DKS[3] and Koorde[11].
2. The choice of the *responsible* node for each portion of the virtual space. The choice of responsible node goes along with the routing mechanism to be used and can be guided by for instance, proximity at the underlying physical network. This is the case for example in Pastry.

In the rest of this report, we shall use the term *exponential overlay networks* to denote the set of all overlay networks that satisfy the following:

1. The size of the virtual space  $N$  is in  $O(k^L)$ , where  $k \geq 2$  and  $L$  is the diameter of the overlay network.
2. The routing table size is either in  $O(1)$  or in  $O(\log_k(N))$ .

We call *constant degree exponential overlay network*, an exponential overlay network in which each node has a routing table with size in  $O(1)$ .

We call *logarithmic degree exponential overlay network*, an exponential overlay network in which each node maintains a routing table with size in  $O(\log_k(N))$ , where  $k$  is the base of the exponential growth of the overlay network.

Based on the above described framework, we are developing effective families of infrastructures that will ease the construction of robust, cost effective and scalable peer-to-peer applications. In the next section, we shall present the progress made during the year 2003, in improving the DKS family of overlay networks, which was directly derived from our framework.

### 3 Enhancements of the DKS overlay networks

As shown by the first progress report on distributed algorithms and services of the PEPITO project, one of the main contributions in the workpackage 2 has been the development of a family of infrastructures, named DKS,

<sup>1</sup>We are working on a paper that proves this claim.

for peer-to-peer applications. The preliminary design of the DKS had a number of limitations. We addressed some of them during the year 2003. In this section, we summarize the improvements made on the previous design.

In this document, we shall assume the reader is familiar with the terminology used in papers like [3, 17].

### 3.1 Network structure

One of the main characteristics of the DKS overlay networks is that the outdated routing entries are corrected on-the-fly, to avoid unnecessary bandwidth consumption that results from the use of periodic stabilization as it is the case in most of the previous peer-to-peer infrastructures such as Chord, CAN and Pastry.

In a DKS overlay network, a node  $n$  updates its routing table in each of the following situations.

1. *Upon receipt of a regular message.* when a node  $n$  receives a “regular” message, for example a lookup request, from another node  $n'$ , node  $n$  adapts its routing table to account for the existence of  $n'$  if necessary. This will be the case when node  $n$  should have, but does not have node  $n'$  in its routing table.
2. *Upon receipt of a notification.* In a DKS overlay network, when a node  $n$  sends a regular message to another node  $n'$ , node  $n'$  first determines whether node  $n$  used a correct routing entry at the time of the sending operation. If node  $n'$  finds out that node  $n$  did not use a correct routing entry, node  $n'$  sends a notification to node  $n$ . This notification message serves to inform node  $n$  about the fact that it has used an outdated routing entry. In order to help node  $n$  correcting itself, node  $n'$  embeds a candidate node  $c$  that node  $n$  uses to replace  $n'$  in its routing table.

In our previous design, the candidate node sent by  $n'$  was the predecessor of  $n'$ . This has the drawback that the convergence of the overlay network towards the perfect state, in which every node has correct routing entries, might be slow. To overcome this problem, we have introduced a *back-list* component at each node. In the back-list, a node maintains information about a predefined number of its predecessors on the circular identifier space (a ring in which for a given node  $n$ , the successor of  $n$  is the first node met in clockwise direction and the predecessor of  $n$  is the first node met in the counter-clockwise direction). Exploiting the back-list component, the convergence of a DKS overlay network towards a perfect state is made faster, as the candidate node used for correction is now selected from a set of candidates. Though this acceleration of the convergence of the overlay network towards a perfect state can be proven analytically, we validated it by means of simulations. We discuss this point in Section 5.

With the addition of the back-list component, special care had to be taken for managing joins, leaves and failures. This led us to design novel algorithms for handling joins, leaves and failures. We elaborate on the novel algorithm for the join operation in subsection 3.3. In subsection 3.4, we sketch the novel algorithm for leave. Then in subsection 3.5 we explain the core ideas behind the novel algorithms for handling failures.

### 3.2 Stack embedded messages for robust recursive lookup

Depending on the application, lookup in a DKS overlay network can be done in one of the following styles: Recursive, Iterative or Transitive. These are well-known strategies borrowed from name resolution in distributed systems [10].

In the recursive lookup approach, upon receipt of a lookup request, a DKS node, if it cannot resolve the target key, will forward the request to another DKS node that is closer to the successor of the target key. This forwarding process is essentially an *interval routing* process, that continues until the successor of the target key identifier is met. From that point the result (successful or not) is returned on the reverse of the path, along which the request was forwarded, up to the originating node.

The problem with this approach is that when forwarding the result backward, the failure of one link/node of the reverse path will prevent the originating node from receiving the result of its request. To overcome this problem, we proposed a novel recursive lookup algorithm in which each request message carries a stack that serves to trace the path followed by the request during the key resolution process. Therefore, during the backward forwarding phase of the result, the stack is popped to determine to which node the result must be sent back. With this improvement, as long as the originating node remains in the system, it will receive the result of its request. In addition, the information carried by the stack is used for correction-on-use.

The price paid by doing this is an increase in the size of the messages. Since the maximum length of a lookup path is logarithmic, it follows that this increase does not impair the scalability of our design.

### 3.3 A novel algorithm for the join operation

In our preliminary design of the DKS overlay networks, each node had in addition to its routing table, a predecessor pointer and a list of its  $f$  successors. We call this list of successors at a node  $n$ , the *front-list* component of  $n$ . One of the purposes of the front-list is to increase the guarantees, as this is used to achieve effective replication.

Given that in a DKS overlay network, each node now has a routing table, a back-list and a front-list and that the system entirely relies on *correction-on-use* for maintaining routing information, we decided to search for a new algorithm for the join operation. As a result of our investigations, we developed a stabilizing algorithm for managing the join operation. The novel algorithm is stabilizing in the sense that when a new node  $n_j$  join the system, the vicinity (union of nodes in the back-list and in the front-list of the node),  $\mathcal{V}(n_j)$ , of  $n_j$  converges quickly towards the legitimate state if ever disturbances cease within  $\mathcal{V}(n_j)$ . However, the insertion of the node itself is done through a local atomic action, that involves the joining node, its current successor and its current predecessor on the ring. Local atomicity of the join is important, as it avoids *false lookup failures*.

### 3.4 A novel algorithm for the leave operation

Work has been done to provide a novel algorithm for the leave operation. This, to incorporate the maintenance of both the back-list and the front-list of the nodes in the vicinity of the departing node.

The novel algorithm for the leave operation is stabilizing. Indeed, when a node  $n$  wishes to leave the system, the departure of  $n$  is coordinated by the successor of  $n$ . This coordination mainly consists in ensuring that each node in the vicinity of the departing node quickly gets a legitimate back-list and/or a front-list, if ever disturbances cease in the vicinity of the departing node. To achieve this, the coordinator triggers a *local diffusion* mechanism that stops as soon as each node in the vicinity of the coordinator has a legitimate back-list and/or a front-list.

Remote nodes not in the vicinity of the  $n$ , but that are pointing to  $n$ , will realize (actually, suspect) that node  $n$  has left the system when they try to communicate with  $n$ , and this will trigger fault-tolerance mechanism that we discuss in the next sub-section.

### 3.5 Novel algorithms for handling failures

Fault-tolerance is probably one of the most challenging tasks in distributed computing in general and in exponential overlay networks in particular.

A significant step has been taken to develop novel effective mechanisms for handling failures. Currently, we are preparing a publication for the developed algorithms. In this section, we shall only sketch the ideas. Before, doing so, we will first present the failure semantics.

### 3.5.1 Failure semantics

The first step in designing fault-tolerant algorithms is to set up the failure semantics. To introduce our failure semantics, let us consider a simple example. Let  $n$  and  $n'$  be two nodes (processes) that are connected by a communication channel (e.g. a TCP connection). Assume that node  $n$  sends a message MSG to node  $n'$  and that node  $n$  expects to receive a reply RPLY from  $n'$  within  $T$  time units. The following two scenarios can be observed when node  $n$  does not receive a timely RPLY:

1. node  $n$  receives within  $T$  time units, a notification message saying that node  $n'$  has failed. Such a notification will typically be sent by the OS above which node  $n'$  was running. In this case, from the node  $n$  perspective, there is an *accurate failure* situation with respect to  $n'$ . The failure situation from the node  $n$  stand-point is accurate, because node  $n$  has received information it can rely on.
2. node  $n$  does not receive within  $T$  time units, any message from  $n'$  nor from the OS above which  $n'$  is (or was) running. In this case, node  $n$  knows for sure that there is failure situation. However, node  $n$  cannot say for sure what is happening. In this case, we say that from the node  $n$  perspective, there is an *inaccurate failure* situation with respect to  $n'$  (a timeout event).

From the above two scenarios, we distinguish two types of failures: *accurate failure* and *inaccurate failure*. We chose these names to embody the important fact that a node observing a failure situation either has an accurate or an inaccurate information about the observed failure situation.

As far as failure detection is concerned, a node  $n$  detects an accurate failure when it receives a reliable notification. Though this type of failures is rare, they do occur. A node  $n$  detects an inaccurate failure when an expected event does not occur in a timely fashion, i.e. a timeout occurs.

It is worth mentioning that investigations on formal aspects on failure semantics and detection are considered in WP1 of the project. Interactions between WP1 and WP 2 in these issues will continue as planned.

### 3.5.2 The novel approach for handling failures

In our earlier design of the DKS, we provided a simple approach to handle accurate failures only. The proposed approach worked fine. However, the convergence of the network towards perfect state was slow.

In our current design, we handle both accurate and inaccurate failures. The principle used for handling both types of failures is as follows. When a node  $n$  detects the (accurate/inaccurate) failure of a node  $n'$ , node  $n$  triggers a local stabilizing diffusion mechanism within the vicinity of node  $n'$ . This diffusion process has an effect to correct all nodes in the vicinity of  $n'$ , if the failure was accurate. In the case the failure was inaccurate, the local stabilizing diffusion process enables to gather enough information in order to take an appropriate action. The local diffusion is stabilizing in the sense that it stops as soon as all nodes in the vicinity of  $n$  are in “correct” state.

## 3.6 One-to-many communication primitives in DKS

We have enhanced the API provided by the DKS family of overlay networks. One-to-many communication primitives are important for various distributed applications. We started to investigate the broadcast primitive in structured overlay networks during the year 2002. The result of our first attempt was published in [6].

### 3.6.1 Broadcast

During the year 2003, we continued to work on broadcast. The focus has been put on broadcast algorithms capable to cope with outdated routing entries, that are common in structured overlay networks that build upon interval routing and correction-on-use, as it is the case in DKS overlay networks.

We developed two broadcast algorithms that integrate our correction-on-use philosophy. The proposed algorithms have the particularity that they correct outdated routing entries while disseminating broadcast messages to all nodes in the system without redundancy. We call these broadcast algorithms, correcting broadcast algorithms [9].

### 3.6.2 Multicast

Effort has been put in understanding facets of multicast in structured peer-to-peer overlay networks. After a careful analysis of the existing approaches to provide multicast in such overlay networks, we were able to come up with an interesting design that builds upon our work on broadcast.

Briefly, the idea consists of creating for each multicast group, an instance of DKS overlay network. Then, dissemination of multicast messages within a given group can be achieved using any of our correcting broadcasts. Details are given in [4].

Unlike previous attempts to multicast in DHTs, our system does not build a single source tree for each multicast group and does not require non-group members to participate in the dissemination of multicast messages, as in [5]. Furthermore, under normal system operation, our multicast approach does not send redundant messages as in [15]. Our multicast design is flexible, as each multicast group can be tailored to fit specific requirements.

### 3.7 Reducing communication with eager update

In *interval-based* routing overlay networks such as DKS and Chord, communication cost can be further reduced by eagerly updating some routing tables, as soon as a node joins, leaves or fails.

To do this, we must address the following problems:

- ▷ Given a node  $n$  in a DKS/Chord overlay network, what are the nodes that, in a perfect state of the overlay network, are pointing to the node  $n$ . We call these nodes the *dependent nodes* of  $n$  and denote by  $\mathcal{D}(n)$  the set of dependent nodes of  $n$ .
- ▷ How to notify nodes in  $\mathcal{D}(n)$  about a specific event?

During the reported period of this document, work has been done to address these problems. We provide an algorithm for computing exactly the dependent nodes and we use our previous work on broadcast for notification. Currently, a paper is being written for publication of these results.

In one of our next steps, we shall integrate this eager update algorithms into our family of overlay networks, to further reduce communication cost.

## 4 Simulation environment for peer-to-peer algorithms

Designing correct, robust and scalable distributed algorithms is a challenging task, that often requires substantial amount of time. A common approach to increase the designer's confidence is to experiment with simulations, though that is not enough to ensure correctness.

Work has continued to provide a suitable simulation environment for distributed algorithms in large, and for fault-tolerant large-scale peer-to-peer algorithms in particular.

During the first year of the project, inspired by the initial simulation environment from MIT, we urgently developed a working simulation environment. Unfortunately, when we started to perform experiments with large number of nodes, the limitations of our earlier simulation environment became really undesirable. The major problems we had to address were the following:

1. The lack of an efficient mechanism to deal with the selection of identifiers. For instance, how to quickly find a “free” identifier to be assigned to a joining node. Recall, that in all peer-to-peer systems, the assignment of identifiers is assumed to be uniform by using a hash function (typically, SHA 1). Simulating this in face of high dynamism is not a trivial task.
2. The lack of flexibility in generating events in the system. In the previous simulation environment, events were pre-generated using lottery scheduling. This has significant impact on the performance as well.

We addressed these problems and provided effective solutions. For the first problem mentioned, we were able to re-discover the power of the shuffling algorithm ([14], page 318), which we carefully exploited to achieve selection of identifiers in constant time, which is a great improvement when compared to the  $O(N)$  time complexity algorithm that was used in the MIT simulation environment.

In addition to the lottery scheduling previously used for generating events, we now provide implementation of mechanisms that allow generation of events according to the Poisson distribution. Interestingly, the simulation environment is now flexible, as it can be instructed to generate events on-the-fly.

## 5 Simulation-based comparison of Chord versus DKS

Work is being done to compare the Chord system versus the DKS system. Indeed, from the routing tables organization point of view, the Chord system is an instance of DKS when the structuring parameter ( $k$ ) of the DKS is set to 2. However, Chord differs drastically from DKS when it comes to overlay network maintenance. In Chord, active stabilization is used for maintaining routing information. With this approach each node periodically runs some stabilization routines that involve communication for finding correct routing entries. This has the disadvantage that even if the system enters a period of time where the dynamism is not high, the amount of bandwidth consumption used for correcting routing entries is high. In DKS, the correction-on-use technique, the local atomic action for joins and leave are exploited for the maintenance of routing information. With the correction-on-use, outdated routing information is corrected only when needed, thereby reducing the unnecessary bandwidth consumption.

From the maintenance of the overlay networks stand-point, Chord and DKS present two extreme design approaches. We started to investigate by simulations, what could be the best/worst operating environments for these two design approaches. The preliminary results show that our correction-on-use-based design approach outperforms the active stabilization-based design of Chord, when the ratio of the number of lookups over dynamism is high enough. Furthermore, with the introduction of the back-list at each node, our simulation results confirmed our expectations that the use of a back-list at each node will speed up the convergence of a DKS overlay network towards its perfect state. Currently, a paper is being written to publish the results of this comparison.

## 6 Formal specification and verification of DKS’s algorithms

Although not strictly part of WP2, we started work on formal specification and verification of some of the DKS’s algorithms. This work started with a tight collaboration with our colleagues in WP1. We report it here to make the document self-contained.

As a first step in the formal verification of peer-to-peer protocols, we studied the DKS lookup algorithm in a static setting, i.e., where no processes may join or leave the system. We expressed the specification of lookup as well as appropriate portions of the DKS lookup algorithm in a value-passing process calculus, and related the specification and the implementation by weak bisimulation. Interestingly, even though weak bisimilarity

is a priori not sensitive to the number of internal messages (or even divergent behavior), the performance-driven design of the algorithm required us to prove performance properties (maximum number of messages per lookup) in order to obtain the bisimulation.

We now plan to proceed towards a more dynamic setting, where nodes can concurrently join and leave the system, even in concurrency with lookups. Compared to the static case, we will have to treat the local state of the nodes (routing tables, membership status) in a more refined way. Another important issue, common to formal methods for distributed systems, is how to model timeout-based failure detection. Moreover, in some dynamic systems, including DKS, a node that leaves the system according to protocol may still have dangling pointers associated to it, something that is also detected using timeouts. For this reason we will have to treat failure detection even in a model without host failures.

## 7 Symmetric overlay networks and high scalability

Work is in progress to provide symmetric and highly scalable exponential overlay networks.

As shown by the first progress report on distributed algorithms and services, we started work on symmetric structured overlay networks. Symmetry is a useful property in structured peer-to-peer overlay networks, as it can for instance serve to achieve notification in place when the overlay network changes, due to peers voluntarily joining and leaving the system.

Our preliminary investigation of the symmetric exponential overlay networks led us to S-Chord[13]. The S-Chord design has two main limitations. First is its rigidity. Indeed, S-Chord is restricted to the base 2. Therefore, it is not possible to specify an arbitrary base  $k \geq 2$ , according to which routing intervals are going to be built. Second, the scalability of S-Chord is limited, as we shall argue in the next subsection. To overcome these limitations, we have studied how to build parametrized logarithmic exponential overlay networks, assuming relative space division. By parameterizing the base of the exponential growth, one has the possibility to tune the infrastructure according to the application domain. As a result of these investigations, we were able to come up with a design we call Tango. Tango has more than the symmetric aspect, as we describe next.

An interesting question that come in mind concerns the optimality of the network size for a given routing table size in the context of exponential overlay networks. We looked at this problem in the case of logarithmic-degree exponential overlay networks.

By considering systems such as Chord and DKS, in which the relative division of the space is exploited, we have observed that there is an unused path redundancy. Based on this observation, we designed Tango in a way that avoids this problem, thereby providing a higher level of scalability. This work is being prepared for publication.

## 8 Probabilistic broadcast algorithms

Work has continued on probabilistic algorithms. We addressed probabilistic broadcast. The question considered in this setting is how can we disseminate information in a reliable manner among a large set of processes, possibly varying in time, and possibly connected by unreliable channels.

To solve the above mentioned problem, we gave a precise specification of this problem through the notion of Delta-reliable broadcast. Basically, in contrast to what we used to do in LANs, we specify here the reliability of a broadcast in a probabilistic manner and use this notion to compare various broadcast algorithms[7].

Building on our precise specification of the problem we were able to come up with an algorithm to implement a probabilistic broadcast in a completely decentralized manner: no process is supposed to know all processes of the system[8].

## 9 Proof of concepts

To demonstrate the potential of our designs, efforts have been put into the implementation of peer-to-peer infrastructures and on the investigation of some example applications. Experimenting with some example applications appeared to be necessary to enhance the API that is going to be provided by the peer-to-peer infrastructure that will result from the PEPITO project.

### 9.1 Implementation of peer-to-peer infrastructures

Work is being done to provide implementations of the DKS system in Oz as well as in C++. Currently, the Oz based implementation is in a quite advanced state, as our simulation environment is fully implemented in Oz. The current C++ implementation is based on our earlier design, as we urgently made it to study the possibility of integrating DKS's algorithms in the DSS.

In addition, to experiment with the ideas developed in our attempt to incorporate symmetry and provide high scalability in logarithmic exponential overlay networks, we have made a first public release of a peer-to-peer library, called P2PS [1], that implements the Tango algorithm. We will continue to add functionality to this library throughout 2004.

### 9.2 Example applications

#### 9.2.1 DKS-based WebLog application

In an internal project at SICS, an implementation of the DKS within the DSS is being used for the development of a DKS-based WebLog application. The feedback obtained from these interactions with people in workpackage 3 has been quite fruitful, and these interactions between workpackages 2, 3 and 4 will continue the next year as planned.

#### 9.2.2 DKS-Based file system

Work is being done in the design and the implementation of a DKS-based peer-to-peer file system. A design has been made and a C++ based implementation has started. The work is being done by two master students at KTH, in the context of the implementation of the DKS infrastructure.

## 10 Cooperation with other workpackages in PEPITO

### 10.1 Cooperation with WP 1

Closed cooperation with workpackage 1 is going on for formal specification and verification of some of the algorithms developed in workpackage 2. As mentioned in Section 6, we have already interesting work in this direction, though the most challenging part (dealing with dynamism) of the work remains to be done. This cooperation will continue the next year.

### 10.2 Cooperation with WP 3

A tight collaboration with people in WP 3 has started. Work has been done for the integration of DKS's algorithms into the DSS (reported in WP 3). We mainly studied the DSS to investigate the possibility of separating the communication layer of the DSS from its consistency protocols. By doing this work, we were able to observe that the communication layer provided by the DSS is a good candidate for the DKS infrastructure and that the DKS's algorithms fit well in the DSS. This collaboration will continue next year, as planned.

## 11 Information dissemination

The work done in WP 2 during the period covered by this report has led to a number of documents. Some of them have been published and some are under preparation for publication.

### 11.1 Published papers

1. Luc Onana Alima, Ali Ghodsi, Per Brand and Seif Haridi. “Multicast in DKS(N, k, f) Overlay Networks”. Regular paper in the *Proceedings of the 7th International Conference on Principles of Distributed Systems (OPODIS 2003)*, December 10-13, La Martinique, France.

This paper is an elaborated version of the poster paper.

2. Ali Ghodsi, Luc Onana Alima, Sameh El-Ansary, Per Brand and Seif Haridi. “Self-Correcting Broadcast in Distributed Hash Tables”. *Proceedings of the 15th IASTED International Conference Parallel and Distributed Computing and Systems (PDCS)*, November 3-5, 2003 Marina del Rey, CA, USA.

In the first progress report, we mentioned the result of our first attempt to do broadcast in DHT-based overlay networks. It has become apparent that our first broadcast algorithm that appear in[6], was not suitable to cope with outdated routing information that are common in DKS overlay networks.

This paper presents two broadcast algorithms that are suitable for DHT-based overlay networks designed with correction-on-use in mind. The presented algorithms are very effective in that they allow to disseminate message in an optimal way while correcting outdated routing information on-the-fly.

3. Luc Onana Alima, Ali Ghodsi, Sameh El-Ansary, Per Brand and Seif Haridi. “Multicast in DKS(N, k, f) Overlay Networks”. Poster paper in the *Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P2003)*, September 1-3, Linköping, Sweden.

This is a poster paper that presents our preliminary idea on how to achieve multicast in DKS overlay networks.

4. P. Eugster, R. Guerraoui, and P. Kouznetsov, “Delta-Reliable Broadcast”, IEEE International Conference on Distributed Computing (ICDCS’04), to appear.

This paper is a revised version of a technical report that was mentioned in the first progress report. It gives a precise specification of the probabilistic broadcast problem through the notion of Delta-reliable broadcast. Furthermore, it provides a comparison of various broadcast algorithms.

5. P. Eugster, R. Guerraoui, S. Handurukande, A-M Kermarec and P. Kouznetsov, “Lightweight Probabilistic Broadcast”, ACM Transactions on Computer Systems.

This paper gives an algorithm to implement a probabilistic broadcast in a completely decentralized manner: no process is supposed to know all the processes in the system.

### 11.2 Papers in preparation for publication

Draft versions of technical reports are available for our work on:

1. Reducing Communication with Eager Update in DHT-based P2P Networks.
2. A Simple and Efficient Simulation Environment for P2P Systems.
3. A Simulation-based Comparison of Chord versus DKS

4. Fault-tolerance in Correction-on-use based Overlay Networks.
5. Symmetry and high scalability in logarithmic degree exponential overlay networks.

## 12 Attached papers

The following papers are attached to this report.

1. Ali Ghodsi, Luc Onana Alima, Sameh El-Ansary, Per Brand and Seif Haridi. “Self-Correcting Broadcast in Distributed Hash Tables”. *Proceedings of the 15th IASTED International Conference Parallel and Distributed Computing and Systems (PDCS)*, November 3-5, 2003 Marina del Rey, CA, USA.
2. Luc Onana Alima, Ali Ghodsi, Per Brand and Seif Haridi. “Multicast in DKS(N, k, f) Overlay Networks”. Regular paper in the *Proceedings of the /th International Conference on Principles of Distributed Systems (OPODIS 2003)*, December 10-13, La Martinique, France.
3. P. Eugster, R. Guerraoui, and P. Kouznetsov, “Delta-Reliable Broadcast”, IEEE International Conference on Distributed Computing (ICDCS’04), to appear.
4. P. Eugster, R. Guerraoui, S. Handurukande, A-M Kermarec and P. Kouznetsov, “Lightweight Probabilistic Broadcast”, ACM Transactions on Computer Systems, Vol. 21, NO. 4, November 2003, pages 341-374.

## References

- [1] P2PS, 2003. <http://www.mozart-oz.org/mogul/info/cetic-ucl/p2ps.html>.
- [2] Karl Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Lecture Notes in Computer Science*, 2172:179–194, 2001.
- [3] Luc Onana Alima, Sameh El-Ansary, Per Brand, and Seif Haridi. DKS(N, k, f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications. In *The 3rd International Workshop On Global and Peer-To-Peer Computing on Large Scale Distributed Systems-CCGRID2003*, Tokyo, Japan, May 2003. <http://www.ccgrid.org/ccgrid2003>.
- [4] Luc Onana Alima, Ali Ghodsi, Per Brand, and Seif Haridi. Multicast in DKS(N, k, f) Overlay Networks. In *Proceedings of the 7th International Conference on Principles of Distributed Systems (OPODIS 2003)*, La Martinique, France, December 10-13 2003.
- [5] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. SCRIBE: A Large-Scale And Decentralised Application-Level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, 2002.
- [6] Sameh El-Ansary, Luc Onana Alima, Per Brand, and Seif Haridi. Efficient Broadcast in Structured P2P Networks. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, February 2003.
- [7] P. Eugster, R. Guerraoui, and P. Kouznetsov. Delta-Reliable Broadcast. In *Proceedings of the IEEE International Conference on Distributed Computing '04 (to appear)*, 2004.
- [8] P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. *ACM Transaction on Computer Systems*, 21:341–374, November 2004.
- [9] Ali Ghodsi, Luc Onana Alima, Sameh El-Ansary, Per Brand, and Seif Haridi. Self-Correcting Broadcast in Distributed Hash Tables. In *Proceedings of the 15th IASTED International Conference Parallel and Distributed Computing and Systems (PDCS)*, Marina del Rey, CA, USA, November 3-5 2003.
- [10] A. Goscinski. *Distributed Operating Systems, The Logical Design*. Addison-Wesley, 1991.
- [11] M. Frans Kaashoek and David R. Karger. Koorde: A Simple Degree-Optimal Distributed Hash Table. In *Proceedings of the 2nd International Workshop, IPTPS 2003*, pages 98–107, Berkeley, February 2003.
- [12] Petar Maymounkov and David Mazières. Kademia: A Peer-to-peer Information System Based on the XOR Metric. In *The 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002. <http://www.cs.rice.edu/Conferences/IPTPS02/>.
- [13] Valentin Mesaros, Bruno Carton, and Peter Van Roy. S-Chord: Using Symmetry to Improve Lookup Efficiency in Chord. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Las-Vegas, Nevada, USA, June 23-26 2003.
- [14] Jean-Claude Radix. *Pratique Moderne des Probabilités*. Lavoisier - TEC & DOC, 1991.
- [15] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level Multicast using Content-Addressable Networks. In *Third International Workshop on Networked Group Communication (NGC '01)*, 2001. <http://www-mice.cs.ucl.ac.uk/ngc2001/>.

- [16] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Computer Science*, 2218, 2001. [citeseer.nj.nec.com/rowstron01pastry.html](http://citeseer.nj.nec.com/rowstron01pastry.html).
- [17] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, pages 149–160, San Deigo, CA, August 2001.
- [18] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. U. C. Berkeley Technical Report UCB//CSD-01-1141, April 2000.