



PEPITO
IST-2001-33234
PEer-toPeer Implementation and TheOry

Deliverable no: D2.2

**Report on Diffusion Algorithms for
Peer-to-Peer Computing**
(covering 2002.01.01 – 2005.04.30)

REPORT VERSION: first

REPORT PREPARATION DATE: 11th March 2005

CLASSIFICATION: Private

DELIVERABLE NO: D2.2 DUE DATE: Month 38 DELIVERY DATE: Month 38

PROJECT START DATE: 2002.01.01 PROJECT DURATION: 38 months

RESPONSIBLE PARTNER: KTH Stockholm

PARTICIPATING PARTNERS: KTH Stockholm, SICS Sweden

PROJECT COORDINATOR: Swedish Institute of Computer Science AB

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



**Project funded by the European Community under
the 'Information Society Technologies' Programme
(1998–2002)**

Contents

1	Introduction	3
1.1	Objectives	3
2	Diffusing Computations	3
3	Diffusion Based Broadcast in DKS	3
4	DKS Algorithms using Diffusion Based Broadcast	4
4.1	Topology Maintenance	4
4.2	Replication	5
5	Conclusion	5

1 Introduction

This report, D2.2, is a report on diffusion algorithms for peer-to-peer computing.

We proceed as follows. In Section 1.1, we describe the objectives of using diffusion based algorithms in workpackage 2. In Section 2, we recall the what a diffusing computation is. Then, in Section 3, we describe the basic diffusion algorithm used to disseminate messages to nodes in the peer-to-peer overlay. In Section 4 we list algorithms which use the algorithm presented in Section 3 to achieve fault-tolerance, replication, and messaging.

1.1 Objectives

From the beginning of the PEPITO project, one of the goals was to make effective use of diffusion-based algorithms for peer-to-peer systems. As large-scale peer-to-peer systems are dynamic, the goal has been to use diffusion computation to diffuse information to relevant peers. To be precise, the algorithms deployed in the peer-to-peer overlay network should be:

- ▷ Decentralized
- ▷ Fault-tolerant
- ▷ Scalable

2 Diffusing Computations

The term *Diffusion Computation* was invented by Dijkstra and Scholten in their seminal paper [2]. They called a computation diffusing when nodes received information from another node (predecessor), and sent it to all their neighbors (successors). The model can be relaxed to cover for cases where a node receiving a message sends it to a subset of its neighbors. The authors suggested that such an algorithm could be augmented with a termination detection algorithm which would detect when all nodes in the system were waiting for messages, and no messages were in transit in the network. In summary, the algorithm in [2] constructs a spanning tree on the fly that shrinks and grows by broadcasting a message over all existing links of every node, and using a special marker message (signals) to avoid cycles.

The work in [2] is important as it assumes very little about the underlying graph which represents the network. Hence, it is very flexible and suitable for dynamic environments where the number of nodes is constantly varying. It was envisioned early in the PEPITO project that such algorithms should be useful in peer-to-peer overlay systems, which are characterized by high churn, and decentralized structure.

3 Diffusion Based Broadcast in DKS

Highly influenced by the work of Dijkstra and Scholten, work package 2 designed a diffusing broadcast algorithm as reported in [4, 6].

The first version of the algorithm, as appeared in [4], implicitly constructs a spanning tree on-top of a structured peer-to-peer system, and broadcasts arbitrary messages to all nodes in the peer-to-peer overlay. By constructing a spanning tree, redundant messages could be completely avoided, something that troubled the first generation of broadcast-based peer-to-peer systems such as Gnutella and Freenet[9, 5].

The problem of broadcasting over a structured peer-to-peer system proved more difficult than initially expected, as nodes in a peer-to-peer system can have highly outdated routing information, which will result in nodes not being covered by the broadcast. By 2003, an initial version of DKS[3] (see Deliverable 2.9) was available, which used local atomic actions to let nodes join and leave the network. Nodes in the overlay network could still have outdated routing entries, but as a consequence of the local atomic join and leave operations in DKS, nodes in the system formed a ring where the successor and predecessor pointers were correct with high probability.

In [6], the previous broadcast algorithm was further extended to work by completely diffusing its messages to every peer in the spanning tree, but parts of the tree that were unavailable to a node as a result of incorrect routing entries would be delegated to other nodes closer to the destination. Hence, a broadcast message would, by the correctness of the successor and predecessor pointers, reach nodes that initially did not appear in the routing table of other nodes. The broadcast algorithm would also correct incorrect routing entries on the fly, to help the system converge to a legitimate state.

The diffusing broadcast algorithms developed by work package 2 proved to be fundamental to the design of the rest of the peer-to-peer system as is reported in the next section.

4 DKS Algorithms using Diffusion Based Broadcast

The self-correcting broadcast reported in [6] was originally intended to be used to broadcast a message to all nodes in the system. In [1], we used the broadcast algorithm to create groups of structured overlay networks, each representing a multicast group, and using the diffusing broadcast algorithm to diffuse broadcast messages to all nodes within a multicast group.

Broadcasting to all nodes in a peer-to-peer group might be applicable if the system consists of many small multicast groups. A diffusing broadcast algorithm that requires $\Theta(N)$ messages for N nodes will, however, be costly as N grows indefinitely. We therefore designed a *restricted* self-correcting broadcast that diffuses a message to all nodes within a range in the peer-to-peer system. The algorithm still constructs a spanning tree on-the-fly, but only covers a pre-specified part of the system.

A restricted broadcast algorithm proved fruitful in many algorithms as briefly explained.

4.1 Topology Maintenance

The broadcast algorithm explained in the previous sections diffuses messages by using overlay routing information. Hence, it requires an overlay network with routing tables already in place. This routing information has to be maintained by a topology maintenance algorithm in the overlay network.

Most systems use a periodic stabilization protocol to periodically probe their neighbors. In DKS, whenever a node joins, leaves, or fails, the logical determinism in the underlying peer-to-peer system is exploited to identify ranges of logical addresses which should be notified to maintain every nodes routing information correct. The restricted broadcast proved useful, as it could be used to efficiently send a message to all nodes identified by the topology maintenance algorithm. The results of this appeared in [8], which was awarded with a best paper award.

4.2 Replication

Most structured overlay networks today provide a distributed hash table (DHT) abstraction, which essentially is an ordinary hash table partitioned and distributed onto the nodes in the peer-to-peer overlay. To provide fault-tolerance, items in the hash table are replicated in the system. Upon a failure, a copy can thus be fetched from a replica storing the item.

In DKS, replicas are spread out in the system to shield replicas from each other [7]. This way, neighboring nodes in the peer-to-peer system will not be able to affect nearby replicas. Furthermore, the replicas of the same item are completely independent, which enables efficient load-balancing schemes. One problem that occurs is that ungraceful failures require fetching items from other peers to maintain the replication factor. The nodes storing these items are placed in predetermined ranges of identifiers in the overlay. To efficiently fetch all items and hence maintain the replication degree, a fetch message is diffused in the range by using the restricted broadcast algorithm explained in Section 3. The algorithm thus efficiently fetches items in $O(\log(N))$ steps, where N is the number of nodes storing the replicas of items originally belonging to a failed node.

5 Conclusion

The objective of PEPITO to construct fault-tolerant, dynamic, scalable, and decentralized algorithms by using diffusion computation has been achieved. Inspired by the seminal work of Dijkstra and Scholten, a broadcast algorithm was designed which constructed a spanning tree on-top of the dynamic overlay and diffused messages onto it.

The diffusing broadcast algorithm proved useful in making the system fault-tolerant, as it is the basis of the topology maintenance and replication algorithms of DKS. It is also the basis of the application-level multicast provided by DKS.

References

- [1] L. O. Alima, A. Ghodsi, P. Brand, and S. Haridi. Multicast in DKS(N, k, f) Overlay Networks. In *7th International Conference on Principles of Distributed Systems (OPODIS)*, La Martinique, France, December 2003.
- [2] E. W. Dijkstra and C. S. Scholten. Termination Detection for Diffusing Computations. *Information Processing Letters*, 11-1:1-4, 1980.
- [3] Distributed k-ary System. <http://dks.sics.se>, 2005.
- [4] S. El-Ansary, L. O. Alima, P. Brand, and S. Haridi. Efficient Broadcast in Structured P2P Networks. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, February 2003.
- [5] FreeNet. <http://freenet.sourceforge.net>, 2003.
- [6] A. Ghodsi, L. O. Alima, S. El-Ansary, P. Brand, and S. Haridi. Self-Correcting Broadcast in Distributed Hash Tables. In *15th IASTED International Conference, Parallel and Distributed Computing and Systems*, Marina del Rey, CA, USA, November 2003.

- [7] A. Ghodsi, L. O. Alima, and S. Haridi. A Symmetric Replication Scheme for Increased Security and Performance in Structured Overlay Networks. Technical Report TR-2004-11, SICS, May 2004.
- [8] A. Ghodsi, L. O. Alima, and S. Haridi. Low-Bandwidth Topology Maintenance for Robustness in Structured Overlay Networks. In *The 38th HICSS Conference*. Springer Verlag, 2005.
- [9] Gnutella. <http://www.gnutella.com>, 2003.