Efficient Broadcast in Structured P2P Networks

Sameh El-Ansary[1], Luc Onana Alima[2], Per Brand[1], Seif Haridi[2]
[1]Swedish Institute of Computer Science, Kista, Sweden
[2]IMIT-Royal Institute of Technology, Kista, Sweden
{sameh, perbrand}@sics.se, {onana, seif}@it.kth.se *

Abstract

In this position paper, we present an efficient algorithm for performing a broadcast operation with minimal cost in structured DHT-based P2P networks. In a system of N nodes, a broadcast message originating at an arbitrary node reaches all other nodes after exactly $N-1$ messages. We emphasize the perception of a class of DHT systems as a form of distributed k-ary search and we take advantage of that perception in constructing a spanning tree that is utilized for efficient broadcasting. We consider broadcasting as a basic service that adds to existing DHTs the ability to search using arbitrary queries as well as dissiminate/collect global information.

1 Introduction

Research in P2P systems resulted in the creation of many Data/Resource- location systems. Two approaches were used to tackle this problem; the flooding approach and the Distributed Hash Table approach approach. The common characteristic of both approaches is the construction of an application-level overlay network. Table 1 includes some of the major differences between the two approaches.

	Flooding	DHT
Queries	Arbitrary	Key Lookup
Query-Induced Traffic	$O(N)$	$O(log(N))$
Hit Guarantees	Low	High
Connectivity Graph	Random	Structured

Table 1: Flooding Approach vs. DHT Approach

The DHT approach with a structured overlay network, determinism, relatively low traffic and high guarantees is currently perceived in the P2P research community as the "reasonable" approach. Many systems were constructed based on that approach such as Tapestry [15], Pastry [11], CAN [8], Chord [12], Kademlia [7]. In contrast, the flooding-based approach represented by [4, 3] is mainly considered as unscalable based on a number of traffic analyses such as [6, 10].

A missing feature in most DHTs is the ability to perform search based on an arbitrary query rather than key lookups. Extensions to existing DHTs are needed to supply this feature. Arbitrary querying is realized in flooding-based systems via broadcasting. However, the random nature of the overlay network renders the solution costly and with low guarantees.

In this position paper, we show the status of our work on extending DHTs with an efficient broadcast layer. We are primarily investigating how to take advantage of the structured nature of the DHT overlay network in performing efficient broadcasts. We provide broadcasting as a basic service in DHTs that should be be deployed for any kind of global dissemination/collection of data.

In the next section, we describe related work. In section 3, we explain our approach based on the perception of a class of DHTs as systems performing distributed k-ary search. In section 4, we present a broadcast algorithm for one of the DHTs, namely Chord. Some preliminary simulations results are presented in section 6. Finally, we conclude and show intented future work in 7.

2 Related Work

Our work can be classified as an arbitrary-search-supporting extension to DHTs. From that perspective, the following research shares the same goal:
Complex Queries in DHT. In [5], an extension to existing DHT systems was suggested to add the ability of performing complex queries. The approach constructs search indices that enable the performance of database-like queries. This approach differs from ours in that we do not add extra indexing to the DHT. The analysis of the cost of construction, maintenance, and performing join operations is not present at the time of writing of this paper.

As broadcast is a form of group communication, one can classify this work as group communication support for DHTs, in that case the following research is of relevance:
Multicast. The work in [13, 9] addressed the issue of multicast in structured P2P networks. We are not aware,

*This work is funded by the Swedish funding agency VINNOVA, PPC project and by the European IST-FET PEPITO project.

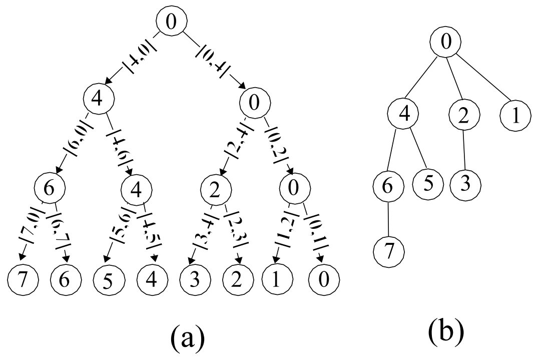

Figure 1: (a) Decision tree for a query originating at node 0 in a fully-populated 8-node Chord network (binary search). (b) The same decision tree without virtual hops, also a spanning tree covering the whole system

though, of any research that tackles the issue of broadcast in DHT-based, sturcuted P2P networks.

3 Our Approach

3.1 DHTs as distributed k-ary search

By looking at the class of DHT systems that have logarithmic performance bounds such as Chord, Tapestry, Pastry, and Kademlia, one can observe that the basic principle behind their operation is performing a form of distributed k-ary search. In the case of Chord, binary search is performed. For other systems like, e.g., Tapestry and Pastry, the search arity is higher.

In this paper, we explain the perception of the Chord system as a special case of distributed k-ary search The arguments apply to higher search arities as well.

The familiarity of the reader to the Chord system and its terminology is assumed. However, we restate the structure of the routing tables. Every Chord node has an identifier that represents its position in a cirular identifier space of size N. A node keeps $M = log_2(N)$ routing entries, called the fingers. A $Finger[i]$ of node n contains the address of a node whose identifier equals $n+2^{i-1}$. If such a node does not exist, then its first succesor in the circular space, moving in a clock-wise direction, is saved instead.

To illustrate the idea of k-ary search, without loss of generality, we assume a Chord system with identifier space of size $N = 8$. The system is fully populated, i.e., a node is present for every identifier in the space. In figure 1.a, we show the decision tree of a lookup query originating at node 0. Given a query for a key whose identifier is x, node 0, starts to lookup for the node responsible for x by considering all the identifier as candidates. Based on the interval to which x belongs (arc labels in figure 1.a), the query is forwarded and the process is repeated with the search scope reduced to a half of the previous scope. Hence, all nodes are reachable by a query-guided path of at most $H = log_2(N)$ hops.

Notice that some of the hops are made from one node to itself, i.e, virtual hops. Figure 1.b shows the same tree decision tree without virtual hops. A more elaborate explanation on the perception of Chord as k-ary search is presented in [2].

3.2 Problem Definition

Having highlighted the idea of distributed k-ary search, we give the following problem definition and then provide a solution for that problem based on the k-ary search perception.

Problem. *Given an overlay network constructed by a P2P DHT system, find an efficient algorithm for broadcasting messages. The algorithm must not depend on global knowledge of membership and must be of equal cost for any member in the system.*

Note that in the problem definition, we emphasize the P2P assumptions, i.e. the absence of central coordination and where every peer endures the same cost for running the algorithm.

3.3 Our Solution

We base our solution on the fact that the k-ary search decision tree is actually a spanning tree. covering all the nodes in the system. Figure 1.b shows the k-ary search decision tree for the 8 nodes system, which is also a spanning tree of the system. In the next section, we show how to construct this tree in a distributed fashion.

4 The Broadcast Algrotihm

4.1 System Model & Notation

We assume a distributed system modeled by a set of nodes communicating by message passing through a communication network that is: (*i*) Connected, (*ii*) Asynchronous, (*ii*) Reliable, and (*iii*) providing FIFO communication.

A distributed algorithm running on a node of the system is described using rules of the form:

$$\frac{\textbf{receive}(Sender : Receiver : \textsc{Message}(arg_1, .., arg_n))}{\text{Action(s) ....}}$$

The rule describes the event of receiving a message $Message$ at the $Receiver$ node and the action(s) taken to handle that event. A $Sender$ of a message executes the statement **send**($Sender$: $Receiver$: $\textsc{Message}(arg_1, .., arg_n)$) to send a message to $Receiver$.

4.2 Rules

Initiating a broadcast. A broadcast is initiated at any node as a result of a user-level request. That is, a user-level layer entity P can send to a node Q a message InitBroadcast($Info$) where $Info$ is a a piece of information that must be broadcast e.g. an arbitrary search query, a statistics gathering query, a notification, etc..

The role of the node Q is to act as a root for a spanning tree. As shown in the rule in figure 2, Q does that by sending a Broadcast message to all its neighbors. Note that, unless the identifier space is fully populated, a Chord Finger table contains many redundant fingers. For a sequence of redundant fingers, the last one is used for forwarding while the others are skipped.

A Broadcast message contains the $Info$ to be broadcast and a $Limit$ argument. A $Limit$ is used to restrict the forwarding space of a receiving node. The $Limit$ of a $Finger[i]$ is $Finger[i+1]$, ($1 \leq i \leq M-1$) where M is the number of entries of the routing table. The last finger's limit is a special case where the $Limit$ is set to the forwarder's identifier. To give an example, we use the sample Chord network given in section 3.1. When node 0 initiates a broadcast, it wants to spread it in the whole identifier space. It sends to nodes 1, 2, and 4. Giving them the limits of 2, 4, and 0 respectively. By doing that it is actually telling node 4 to cover the interval $[4, 0[$, i.e. half of the space. It is telling node 4 to cover the interval $[2, 4[$, i.e., quarter of space and finally, telling node 1 to cover the interval $[1, 2[$, i.e. an eighth of the space.

```
receive($P$ : $Q$ : InitBroadcast($Info$))
for $i$ in 1 to $M-1$ do
  //Skip a redundant finger
  if $Finger[i] \neq Finger[i+1]$
    $R := Finger[i]$
    $Limit := Finger[i+1]$
    send($Q$:$R$:Broadcast($Info$, $Limit$))
  end
end
//Process the last finger
send($Q$:$Finger[M]$:Broadcast($Info$, $Q$))
```

Figure 2: Initiating a broadcast message

Processing a broadcast. A node Q receiving a Broadcast($Info, Limit$) message is responsible for the broadcast in a subtree confined in the interval $]Q, Limit[$. In addition to skipping the redundant fingers, Q forwards to every finger whose identifier is before the $Limit$. Moreover, when forwarding to any finger, it supplies it with a $NewLimit$, defining a smaller subtree. Note that, this will only happen if $NewLimit \in]Q, Limit[$, i.e., the $NewLimit$ is not exceeding the $Limit$ given by the parent. If the $NewLimit$ exceeds $Limit$, the $Limit$ given by the parent is used instead. The following figure contains the rule for processing a broadcast message.

```
receive($P$ : $Q$ : Broadcast($Info$, $Limit$))
for $i$ in 1 to $M-1$ do
  //Skip a redundant finger
  if $Finger[i] \neq Finger[i+1]$ then
    //Forward while within "Limit"
    if $Finger[i] \in ]Q, Limit[$ then
      R := Finger[i]
      //NewLimit must not exceed Limit
      if $Finger[i+1] \in ]Q, Limit[$ then
        $NewLimit := Finger[i+1]$
      else
        $NewLimit := Limit$
      fi
        send($Q$:$R$:Broadcast($Info$,$NewLimit$))
      fi
    else
      exit for
    fi
  fi
end
//Process the last finger
if $Finger[M] \in ]Q, Limit[$ then
  send($Q$:$Finger[M]$:broadcast($Info$, $Q$))
fi
```

Figure 3: Processing a Broadcast Message

Replies. We are considering the issue of replying to the broadcast source to be an orthogonal issue that depends on the $Info$ argument of the Broadcast message. Several strategies could be considered for replying, for example : (i) Sending the broadcast source with every broadcast message and it is contacted directly by a node willing to reply (ii) The reply is propagated to the root over the same spanning tree.

4.3 Correctness Argument

Coverage of all nodes. As a DHT system constructs a *connected* graph of nodes and as every node that receives a broadcast message forwards it to all of its neighbors (except those it knows by DHT construction properties that they are going to be contacted by other nodes), therefore, *eventually* every node in the system receives the broadcast message.

No redundancy. Since the algorithm ensures that disjoint (non-overlapping) intervals are considered for forwarding. Consequently every node receives the broadcast message exactly once.

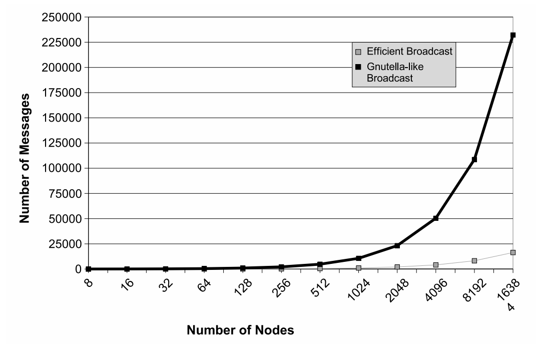

Figure 4: Number of messages needed to cover every node in the system using the efficient broadcast and the Gnutella algorithms.

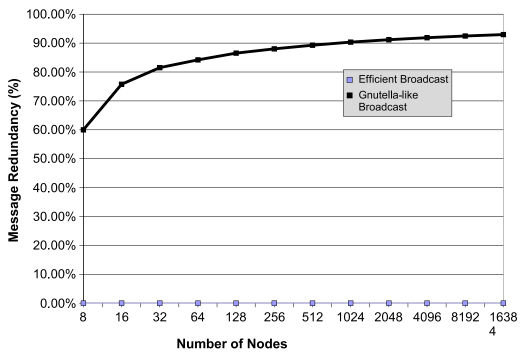

Figure 5: Percentage of redundant messages generated by the efficient broadcast and the Gnutella algorithms.

5 Cost Versus Guarantees

While presenting an efficient algorithm for broadcast in DHT-based P2P networks, we are aware that the cost of N-1 messages, especially in large P2P systems can be prohibitive for many applications. The point is that we offer broadcasting as a basic service available for a Peer who is willing to endure its cost. Our algorithm offers strong guarantees and utilization of traffic for that endured cost. In order to offer the same guarantees on a network, of the same size, in a Gnutella-like broadcast, a substantially higher cost is paid. The next section elaborates more on this comparison.

Predictable Guarantees. The broadcast as presented in section 4, offers strong guarantees as it explores every node in the network. Minor modifications to the algorithm could be applied to, deterministically, reduce the scope of the broadcast, and thus offering weaker, yet predictable guarantees. For example, by sending only to the last (or all but the last) finger while initiating a broadcast, only 50% of the network is covered in the broadcast. Similar pruning policies could be applied to achieve different coverage percentages.

Different Traversal Policies. The algorithm could also be modified to support and iterative deepening policy. This policy was suggested in [14] for use in unstructured overlay networks. We believe that combining this policy with our algorithm can decrease the messaging cost, especially, when one query hit suffices as a result.

6 Simulation Results

In this section, we show preliminary simulation results for the presented broadcast algorithm. We are primarily interested to see that all nodes are covered in the broadcast process and that no redundant messages are sent. Additionally, we want to compare the messaging cost of the efficient broadcast algorithm with that of the Gnutella broadcast algorithm over the same size of the network and with the same guarantees offered. The experiments were conducted on a distributed algorithms simulator developed at the Swedish Institute of Computer Science and using the Mozart [1] programming platform.

Experiments Setting. To study the messaging cost, we create an identifier space of size $2^{16}$and we vary the number of nodes in the space, from 2^3 up to 2^{14} with increasing powers of 2. For each network size, after all the nodes join the system, we initiate a broadcast process starting at a randomly-chosen node. We wait until the broadcast process ends and, then, analyze the messages to see if all the nodes are covered and count the amount of redundant messages. We repeat the same experiment a number of times, initiating the broadcast from different sources.

Both the efficient and the Gnutella algorithms are evaluated in the same way. We use the basic Gnutella algorithm except that we deploy it on a structured rather than a randomly-connected overlay network. That is, the unique fingers of the Chord nodes are used as neighbors. Moreover, we set the Time-To-Live (TTL) parameter of the Gnutella broadcast to the diameter of the network , i.e. $log_2(N)$ which should be just enough to guarantee that all the nodes of the network are covered.

Results. For the number of messages, the efficient broadcast algorithm constantly produces N-1 messages for the different network sizes. The Gnutella algorithm succeeds to cover all the nodes, thanks, to the TTL parameter, but does that with a substantially larger amount of messages. The comparison is shown in figure 4. The reason for that difference is the redundant message that are sent in the Gnutella case and are eliminated in the efficient broadcast case. It is worth noting that the amount of redundancy increases with system size, strongly affecting scalability if the strong guarantees are to be maintained. Figure 5 shows the percentage of redundant messages from the total number of messages generated by both algorithms.

7 Conclusion and Future Work

In this paper, we showed the status of our work in extending the functionality of DHTs with the ability to perform efficient broadcasts. Our approach depended mainly on the perception of systems such as Chord, Tapestry, Pastry, and Kademlia as implementations of distributed k-ary search. We showed how to traverse search tree and thus, constructing a spanning tree containing all nodes in an overlay network formed by a DHT.

We based all our explanation on Chord as a simple system implementing binary search. In future papers, we intend to elaborate more on how to construct the spanning tree in systems with higher arities.

We suggested a number of strategies by which a peer deploying the efficient broadcast algorithm can reduce its scope by pruning the spanning tree in order to generate less traffic, yet with the ability to deterministically decide the percentage of network members that are covered in the broadcast and thus offering predictable guarantees. More experiments need to be done for the evaluation of those strategies.

For the issue of dynamic network (Joins/Leaves), we consider that our algorithm, will perform as good as the underlying DHT system. Knowing that the routing tables of nodes in a DHT are constantly changing, more experimental results are needed to quantify the effect of outdated routing tables on the properties offered by the efficient broadcast algorithm.

References

[1] Mozart Consotium, *http://www.mozart-oz.org*.

[2] Sameh El-Ansary, Luc Onana Alima, Per Brand, and Seif Haridi, *A framework for peer-to-peer lookup services based on k-ary search*, Tech. Report TR-2002-06, SICS, May 2002.

[3] FreeNet, *http://freenet.sourceforge.net*.

[4] Gnutella, *http://www.gnutella.com*.

[5] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, and Boon Thau Loo, *Complex queries in dht-based peer-to-peer networks*, The 1st Interational Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.

[6] E. P. Markatos, *Tracing a large-scale peer to peer system: An hour in the life of gnutella*, Second International Symposium on Cluster Computing and the Grid, 2002.

[7] Petar Maymounkov and David Mazires, *Kademlia: A peer-to-peer information system based on the xor metric*, The 1st Interational Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.

[8] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, *A scalable content addressable network*, Tech. Report TR-00-010, Berkeley, CA, 2000.

[9] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker, *Application-level multicast using content-addressable networks*, Third International Workshop on Networked Group Communication (NGC '01), 2001.

[10] M. Ripeanu, I. Foster, and A. Iamnitchi, *Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design*, 2002.

[11] 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).

[12] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, Tech. Report TR-819, MIT, January 2002.

[13] Ion Stoica, Dan Adkins, Sylvia Ratnasamy, Scott Shenker, Sonesh Surana, and Shelley Zhuang, *Internet indirection infrastructure*, The 1st Interational Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.

[14] Beverly Yang and Hector Garcia-Molina, *Efficient search in peer-to-peer networks*, The 22nd International Conference on Distributed Computing Systems (ICDCS 2002), 2001.

[15] 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.