Dynamic TTL-Based Search In Unstructured Peer-to-Peer Networks

Imen Filali, Fabrice Huet

Dynamic TTL-Based Search In Unstructured Peer-to-Peer Networks

Imen Filali and Fabrice Huet
INRIA Sophia-Antipolis, Université de Nice Sophia-Antipolis, CNRS - I3S
2004, Route des Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex, France
{*Imen.Filali, Fabrice.Huet*}*@sophia.inria.fr*

Abstract—**Resource discovery is a challenging issue in unstructured peer-to-peer networks. Blind search approaches, including flooding and random walks, are the two typical algorithms used in such systems. Blind flooding is not scalable because of its high communication cost. On the other hand, the performance of random walks approaches largely depends on the random choice of walks. Some informed mechanisms use additional information, usually obtained from previous queries, for routing. Such approaches can reduce the traffic overhead but they limit the query coverage. Furthermore, they usually rely on complex protocols to maintain information at each peer.**

In this paper, we propose two schemes which can be used to improve the search performance in unstructured peer-to-peer networks. The first one is a simple caching mechanism based on resource descriptions. Peers that offer resources send periodic advertisement messages. These messages are stored into a cache and are used for routing requests. The second scheme is a dynamic Time-To-Live (TTL) enabling messages to break their horizon. Instead of decreasing the query TTL by 1 at each hop, it is decreased by a value v such as $0 < v < 1$. Our aim is not only to redirect queries towards the right direction but also to stimulate them in order to reliably discover rare resources. We then propose a Dynamic Resource Discovery Protocol (DRDP) which uses the two previously described mechanisms. Through extensive simulations, we show that our approach achieves a high success rate while incurring a low search traffic.

Keywords-**Unstructured Peer-to-Peer networks; Resource discovery; Dynamic-TTL; P2P computing**

I. Introduction

Peer-to-Peer (P2P) systems are considered as an efficient paradigm to build large scale distributed applications such as file sharing or distributed computing.

These systems are often classified into two main categories: structured and unstructured P2P systems. Structured P2P overlays (e.g., Chord [1], CAN [2], Pastry [3], etc.) impose strict constraints on network topology and data location whereas unstructured P2P architectures have no such requirement. Due to their simplicity and their high robustness, most of P2P applications operate on unstructured P2P networks. However, one important challenge in such environments is the resource discovery problem: How can a peer *p* that needs a particular resource *r*, efficiently find a node that provides it.

The basic resource location schemes adopted in unstructured P2P environments are usually based on blind search mechanisms such as flooding or random walks [4]. However, these methods are not highly efficient since peers are randomly connected to each other which induce a lot of useless queries propagating over the network.

In pure flooding, a peer sends the query to *all* its neighbors which, in turn, propagate it to *all* their own neighbors, and so on. The query starts with an initial Time-To-Live (TTL) which is decreased by one at each hop. It is discarded whenever the resource is found or the message TTL expires. The farthest peers which can be reached with a given TTL are often referred to as the horizon. As shown in [4] [5], this approach generates a significant amount of messages.

In random walks approaches [4] [6], a query message is forwarded to a randomly chosen neighbor at each step until the resource is found. Although, it cuts down the message overhead significantly, this method decreases the success rate especially when looking for rare resources. A simple proposed improvement is to initially generate k messages rather than only 1. Likewise, the network topology and the random choices made at each forwarding step have a great impact of the performance of this approach. Our approach for resource discovery seeks to retain the simplicity of the two previously described schemes, while greatly improving their performance. In this paper, we propose a *Dynamic Resource Discovery Protocol (DRDP)* which combines a *cache-based approach* with *dynamic TTL strategy* in order to efficiently route queries towards peers offering the requested resources.

Our objectives through this design are to provide a high success rate for queries while still maintaining a low overhead. The resulting protocol is simple, easy to implement and does not require complex signaling and synchronization between peers.

We evaluate our approach, through extensive simulations, in accordance with several performance metrics including the success rate and the search cost. Simulation results show that our mechanism reduces the search cost and increases the success rate.

The contributions of our work are:

- A simple but efficient cache system based on resource description.
- A cache-based routing scheme aiming at forwarding messages towards resource location.
- A dynamic TTL which enables messages to break their horizon when necessary while still maintaining a low

overhead.
- A set of schemes which can be used to improve the performance of existing search protocols based on caching and TTL.

The rest of the paper is organized as follows: Section II presents an overview of search mechanisms in unstructured P2P networks. Section III details the cache-based and the dynamic TTL approaches to improve search performance in unstructured P2P networks. A new protocol, called *DRDP*, is proposed and evaluated in Section IV. Section V concludes the paper and outlines future works.

II. Related Work

A large amount of work carried out in P2P systems and tackle, in particular, the resource discovery problem. As the most existing search algorithms in unstructured P2P networks are flooding-based mechanisms such as in Gnutella [7], a set of studies focus on that problem and propose different enhancements in order to improve search efficiency. Service discovery schemes fall into three main categories: forwarding-based, cache-based and overlay optimization.

In forwarding based mechanisms, instead of sending the message to all its neighbors, a peer selects a subset of neighbors to forward the message to. In *k*-random walks [4] scheme, a query is forwarded to *k* neighbors from the query originator. For each walk, the query is forwarded to one neighbor randomly selected. Because of their blindness property, random walks based methods are non-deterministic. Although having usually a low search cost, the success rate greatly depends on the random choice of neighbors to forward the query to.

The message forwarding strategy proposed in [8] uses feedback from previous searches to probabilistically guide future queries. Similar approaches are proposed in [9] [10] on which queries are also forwarded to a set of neighbors according to statistics of previous queries content. ARPS presented in [11] is another weighted probabilistic forwarding mechanism where queries are forwarded according to the node degree distribution and resource popularity.

Normalized Flooding [12] approach is similar to flooding, except that each node sends the message only to a set of its neighbors. Consider that λ is the minimum degree (i.e., number of neighbors) of a given node. If the node has more than λ neighbors, then it randomly selects only λ neighbors to send the query to. In [13], Reza *et al.* introduce hybrid algorithm that combines flooding with random walks. It uses flooding to find, say k, outer nodes and then starts k independent random walks from these nodes. In the expanding ring [4] query propagation mechanism, the peer forwards the query with a small TTL and waits to see if the search is successful. If it is, the peer stops the flooding. Otherwise, it initiates another lookup with a larger TTL. However, even if it this mechanism can partially solve the TTL selection problem, it does not take into consideration the message duplication issue.

As mentioned before, other service discovery approaches are based on caching mechanisms. In [14], Doulkeridis *et al.* propose an XML based caching approach. XML documents are used to describe resources stored on remote peers and are stored into caches. Searching for a document is performed using flooding and looking in the caches. When a matching document is found, the remote peer is directly contacted, but the flooding continues. Caches are filled when matches are found and decay over time. The main drawback of this approach is that caches are only built during successful searches, introducing a cold start phase. IAC [15] is an Interest-Aware Caching mechanism. Peers can advertise their resources. Only peers interested in these resources can store advertisements in their local caches. When performing a search process, a peer looks in its local cache for an appropriate matching. If a local cache search fails the random walks based search mechanism is used. The Distributed Resource Location Protocol proposed in [16] requires that the collected results have to follow the query path.

Some other solutions have been proposed taking advantage of the network topology in order to enhance the system performance. In GUESS [17], searches are performed by iteratively contact different super-peers that in turn ask their leaf-nodes for the requested object. In Gnutella2 [18], a super-peer that receives a query from a leaf node forwards it to its relevant leaves and to its neighboring super-peers. The latter forwards it to their relevant leaves. In order to reduce the overhead in such hybrid model, the number of leaf-nodes per super-peer must be kept high. This is considered as an important requirement in order to reduce message forwarding cost and increase the number of discovered objects. Other studies (e.g., [19] [20]) dynamically adapt the network topology according to peers' interests.

Overall, several search schemes have been proposed to address the resource discovery issue in unstructured networks. Blind search mechanisms are either not scalable or have a low success rate. Other mechanisms, such as [8] and [15] maintain local information for routing but mostly improve performance for similar queries. Adding super-peer can reduce the search cost, at the expense of a lower reliability, as a super-peer can become a single point of failure for its cluster. It also introduces constraints on the topology, increasing the network maintenance cost.

III. Protocol Design

This section gives an overview of the basic principles of DRDP. It also details the key components used in its design including the cache update strategies and dynamic TTL-based forwarding.

A. Overview

A resource discovery mechanism often involves two major operations: the query routing and the query matching. The goal of DRDP is to provide an efficient message forwarding mechanism in a pure P2P environment. Instead of blindly routing queries to all neighborhood peers, a peer will try to forward a message towards peers that are the most likely to have the requested resource(s). To achieve this purpose, DRDP combines two simple but efficient mechanisms. First, it is based on a *push-pull* model where available resources are advertised through specific messages. Each peer in the network maintains a set of caches, one for each of its links. These caches are used for temporarily storing the received description of resources. This information can later be used to narrow down the number of neighbors to forward the query to and hence find requested resource(s) with a limited number of queries. If no match is found in the cache, DRDP can use either flooding or random walks. Second, we adopt a dynamic TTL-based approach to increase the probability to find a requested resource. This strategy is used whenever a cached resource description matches a request. It can be viewed as a way to stimulate the query to follow the direction from which the matching advertisement came.

From now on, we use will the term *grant* node to refer to a peer that advertises its resources by sending grant messages. A peer looking for resources is referred as *search* node and can generate request messages. However, these roles are not exclusive, since any peer can act as both a grant and search node.

As illustrated in Figure 1, DRDP performs three main tasks: resource advertising (peer p_3), resource requesting (peer p_1) and message redirection (peer p_2).

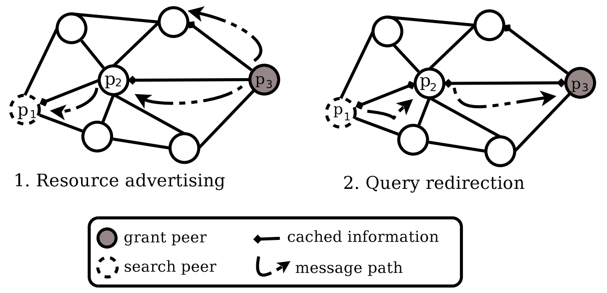


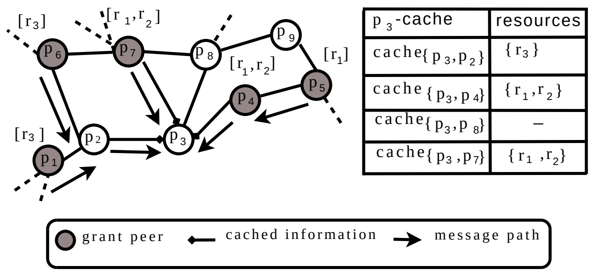
Figure 1. The basic operations of DRDP approach

B. Caching

Each peer has a cache associated to each of its links. As an example, a peer having 5 neighbors will have 5 different caches. An entry in a cache is composed of 2 fields: a timestamp and the description of the advertised or queried resource(s). As opposed to the usual caching schemes, we do not store the originator of the message as we do not perform direct connection. Rather, we try to maintain enough information allowing an efficient message routing.

The main benefit from having a cache per link is that messages pertaining to similar resources, but emitted by different peers, will be aggregated into one single entry. This resource aggregation policy ensures a small cache size. More formally, let $cache_{\{i,j\}}$ be the cache system associated to the incoming link connecting peer i to its neighbor j. Figure 2 illustrates the per-link cache based approach. In particular, it exhibits the per-link cache system associated to peer p_3 (p_3-cache). Arrows represent paths taken by advertisements generated by grant peers (p_1, p_4, p_5, p_6, p_7) before being received by peer p_3.

For instance, peer p_3 has received two advertisements for the same resource $[r_3]$, one generated by p_6 and the other by p_1. As both advertisements are forwarded by the same neighbor p_2, only one entry will be added to $cache_{\{p_3,p_2\}}$. It is possible that a resource description might occupy an entry in more than one per-link cache if related messages (generated by different peers) were routed through more than on link. A simple illustration is also given by Figure 2 where the peer p_3 has received the same resource $[r_1]$ from p_4 and p_7. In this case, one entry will be added in $cache_{\{p_3,p_4\}}$ and the other one in $cache_{\{p_3,p_7\}}$. Using the cache content associated to peer incoming links, each node tries to identify the best possible node either to serve a search peer's request or to better use grant peer resources.



p_3-cache	resources
$cache_{\{p_3,p_2\}}$	$\{r_3\}$
$cache_{\{p_3,p_4\}}$	$\{r_1,r_2\}$
$cache_{\{p_3,p_8\}}$	–
$cache_{\{p_3,p_7\}}$	$\{r_1,r_2\}$

Figure 2. The per-link cache system of the peer p_3

Cached resources description could become out-of-date under resource dynamicity. The longer a resource is cached, the more stale it becomes. Finding a balance between caching and inconsistency is the key to achieve good performance. For this reason, each cached resource description has an associated cache lifetime c_{lt} to limit the occupation and maintain an up-to-date information. If c_{lt} has been expired, the resource will be automatically removed. Moreover, a peer tries to maintain in the cache only the most recent information about advertised (resp. requested) resources. More precisely, whenever a message is received, the peer extracts the description of different resources. For each resource r, it first checks if there is an associated cache entry. If an entry is found, the peer updates this entry. Otherwise, a new one is added to the cache associated to the link from which the current message is received. Thus, a new message will always overwrite previous cache entries. Each cache has a limited size c_s that confines the number of entries per link. Once the cache is full, the First In First Out (FIFO) cache replacement policy is used to keep the most recent entries in the cache. An important property is that the cache size does not depend on the number of peers in the network. Rather, it depends on the number of different resource classes due to resource aggregation policy.

C. DRDP forwarding strategies

There are two forwarding strategies in our protocol, depending on the cache content and the type of the message to be routed. The default strategy relies on blind search approaches and is used when no particular cached information is available. The second is a cache-based routing scheme and is adopted when a matched information is found. Based on the cache content, two routing decisions will be taken: *to which neighbor(s) the query will be relayed to and by which value the current query TTL will be decreased*?

1) Forwarding: When receiving a request message, a peer looks into its caches. When matches are found, the message is forwarded to a randomly selected link among the matching links. If no such link is found, then the message is flooded. Grant messages are forwarded using random walks, except when a match is found in a cache. In that case, they are forwarded like request messages. The random walks ensure a distribution of advertisements at a lower cost than flooding. Therefore, the more advertisements are spread in the network, the better the information in the caches is. When a matching is found, the single forwarding helps propagating the message towards a peer requesting the resource.

2) Dynamic TTL: We adopt two different handling of messages'TTL, based on their type. Grant messages' TTL are always decremented by 1 at each forwarding step. Request messages benefit from a slightly different scheme. When receiving a request, a peer will decrement its TTL based on the cache content. If a matching is (resp. not) found, the TTL is decremented by a value $0 < v < 1$ (resp. 1). The idea is to give an incentive to requests going in the direction of the needed resource.

For better understanding, consider the scenario depicted in Figure 3 which illustrates a query routing example. The TTL value is initially set to 2 and the decrement value v is fixed to 0.5. Peer p_5 acts as a grant node and p_1 as a search peer. Under the same setting, we compare the query routing process using the DRDP approach and the basic flooding mechanism.

We assume that $cache_{\{p_1,p_2\}}$, $cache_{\{p_2,p_3\}}$, $cache_{\{p_3,p_4\}}$ and $cache_{\{p_4,p_5\}}$ hold information about a resource advertised by p_5. Therefore, a query sent by p_1 will get a positive feedback from the caches and thus will have its TTL decreased by 0.5. At each step, it is sent to the neighbor from which the resource advertisement has been received. Indeed, the request message is delivered along the overlay path $(p_1, p_2, p_3, p_4, p_5)$. In this scenario, only four nodes are queried in order to locate the requested resource.

Now, consider the same scenario using a simple flooding mechansim for the query routing. Figure 3 shows that eight nodes are visited along the query search process. Although, in this particular example, flooding generates many more messages than DRDP, it fails to locate the requested resource. This is not surprising since the resource is located at 4 hops from the peer p_1 and the initial TTL value was set to 2. Increasing the initial TTL value will improve the search but also greatly increase the number of messages.

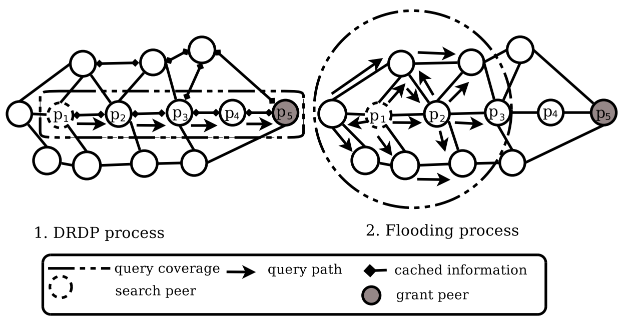


Figure 3. A simple comparison of DRDP and flooding when using the dynamic-TTL forwarding strategy

Before performing a request forwarding process, a peer performs a search operation looking for *the best neighbor*. The best neighbor algorithm records the number of matched resources received from each direction. The request is then sent to the peer that forwarded the largest number of resources. If there is more than one best neighbor, one peer will be randomly selected. This ensures a load balancing by the fact that at each routing operation, message will be randomly send to a neighbor from the subset of the best forwarders. Algorithms 1 and 2 respectively give the best neighbor selection and request forwarding process where we use the following notations:

- $q(f, n, ttl, \mathcal{R} = [r_1, r_2, ..., r_x])$ is a request message, looking for $[r_1, r_2, ..., r_x]$ forwarded by peer f and received by node n.
- v is the TTL decrement value.
- $\mathcal{N}(n)$ is the neighborhood set of peer n. It is defined as *all* peers that are located at exactly one hop from n.

Through the cache system design and the dynamic TTL, it is possible to forward a request message with a $ttl < 1$. If we denote by h_{max} the maximum number of hops that can be performed by a given query, then $h_{max} = \frac{ttl}{v}$. This strongly depends on: $i)$ the cache content, $ii)$ the resource location, in terms of hop distance, from the requester peer.

Table I summarizes the different forwarding strategies used in DRDP.

Algorithm 1 Best Neighbor(s) Selection

```
Require: $q(f, n, ttl, \mathcal{R} = [r_1, r_2, ..., r_x])$, $\{f, i\} \in \mathcal{N}(n)$
Ensure: $\mathcal{BN}$
 1: for $p \in \mathcal{N}(n)$ do
 2:     for $r \in \mathcal{R}$ do
 3:         find $\overline{r}$ that $matches(r)$ forwarded by $p$
 4:         if $\overline{r}$ is found then
 5:             $matchesNbr_q(p) \leftarrow matchesNbr_q + 1$ {increment number of matched resources forwarded by $p$}
 6:         end if
 7:     end for
 8: end for
 9: $\mathcal{BN} \leftarrow Forwarders(Max(\ matchesNbr_q))$
10: return $\mathcal{BN}$
```

Algorithm 2 Dynamic TTL-based forwarding

```
Require: $q(f, n, ttl, \mathcal{R} = [r_1, r_2, ..., r_x])$, $\{f, i\} \in \mathcal{N}(n)$
 1: $\mathcal{BN} \leftarrow BestNeighborSelection(\mathcal{N}(n), q)$
 2: if $\mathcal{BN} \neq \emptyset$ then
 3:     pick up $i$, such as $i \in \mathcal{BN}$
 4:     $ttl \leftarrow ttl - v$
 5:     send $q$ to $i$
 6: else
 7:     $ttl \leftarrow ttl - 1$
 8:     send $q$ to $\mathcal{N}(n)$ except $f$
 9: end if
```

In this section we have presented two simple schemes (caching and dynamic TTL) and various routing policies. Although, as we will see in the next section, they can directly be used to build a specific protocol, we believe they can be integrated in existing one. Any search algorithm based on TTL and informed search can be modified to accommodate a dynamic TTL and improve its performance.

IV. A Dynamic Resource Discovery Protocol and its Evaluation

In this section, we present simulation results to evaluate the performance of DRDP. The first part focuses on a comparative study of the proposed scheme with the flooding and random walks approaches under a set of performance metrics. The second part analyzes the impact of DRDP parameters on its performance. A cycle based simulator [21] was used for both modeling the network topology and simulating the behavior of different approaches in a Grid environment.

A. Context

In the context of Grid system, resources are characterized by their diversity. In such system, users often need to discover software or hardware resources based on extensible set of resource descriptions. When submitting a job, users

Table I
MESSAGE FORWARDING POLICIES

cache content	**grant message**	**request message**
cache matching	- one walk - static TTL	- one walk - dynamic TTL
no cache mathing	- k random walks - static TTL	- flooding - static TTL

has to specify its requirements like memory, disk space, operating system. In the following, we consider a P2P Grid environment with a set of N peers where *grant* peers offer resources such as free disk space, CPU cycles, memory, etc.

A user issues a query q through so called *search* peers and provides a description of the required resources $R = (r_1, r_2, ...r_x)$. It can be a single resource (e.g., storage resource)or a set (e.g, storage resource *AND* a computing resource) of independent resources. When receiving a request, a grant peer looks whether it matches any of its available resources $\overline{R} = (\overline{r_1}, \overline{r_2}, ...\overline{r_y})$. If so, matched resources are removed from the request and an acknowledge message is sent back to the search peer. The query, with the remaining resources, is then forwarded using the chosen scheme (flooding, random walks, DRDP). Note that, in our simulations, all requests are generated such that for each requested resource r there is at least one matching resource $\overline{r}$ provided by the network.

B. Simulation environment

The parameters that determine the simulation scenarios fall into three basic classes: network topology, querying model and resource distribution.

Network model. In the simulation study presented in this paper, we mainly use random graph topologies.

Message distribution. In DRDP approach two basic messages are generated: request messages and grant messages. However, in flooding and random walks approaches only request messages are generated. In all approaches, we use exponential distribution as a message generator model, i.e., the time interval (in simulation cycles) between two consecutive messages generated by a given peer follows an exponential distribution.

Resources. The role of each peer is randomly assigned at start up time based on the *grant nodes* (N_g) and *search nodes* (N_s) parameters. Without any loss of generality, we have used integers to describe resources during the simulation. The number of different resource classes is specified through the *number of resource classes* (r_c). The maximum number of resources that a peer can request in a single query is defined by *number of requested resources per query* (n_{sr}). A search peer sends a query and waits for replies from grant peers. Each grant peer holds a set of resources. This number is randomly selected at initialization time and is specified by *number of resources per grant peer* (n_{gr}). Unless stated otherwise, simulations are based on the default parameters

Table II
SIMULATION PARAMETERS

parameters	default values
network size (N)	10000
peer average degree (avg_d)	6
grant nodes (N_g)	4000
search nodes (N_s)	6000
number of resource classes (r_c)	3000
number of resources per grant peer (n_{gr})	$[10-80]$
number of requested resources per query (n_{sr})	$[1-5]$
mean query generation time (λ)	20000
mean grant generation time (μ)	30000
message cache life time (c_{lt})	30000
maximum cache size (c_s)	1000
TTL value (ttl)	3
DRDP decrement value (v)	0.3
walks number (k)	15
grant walks number (k_g)	15

illustrated in Table II.

In this scenario, 40% of grant peers are selected out from 10000 nodes, each offering a distinct number of resources, n_{gr}, fluctuating between 10 and 80. Resources, distributed across the network, belong to 3000 types of available resource classes, r_c. Let p be the replication ratio for a given resource r. This parameter is defined as the number of resource "replicas" belonging to the same class over the network size N. As an example, for $N = 10000$, $p(r) = 0.001$ means that 10 replicas of r are available in the network. Notice that p has a great impact on the search process, at least in terms of successful searches, as the more replicated a resource is, the higher the probability to find it.

C. Performance metrics

We consider three performance metrics to evaluate the proposed approach: the success rate, the traffic overhead and the cache hit rate.

Success rate. A query is successful if and only if *all* requested resources specified in the query are found. Thus, the success rate, denoted by s, is given by:

$$s = \frac{\sum_{i=1}^{N_s} s_i}{\sum_{i=1}^{N_s} q_i}$$

where s_i, q_i present respectively the number of successful searches and the number of queries generated by peer i.

Traffic overhead. This metric is computed in terms of the total number of queries sent or received by peers in the network during the search process. It includes, grant messages generated when using DRDP. Therefore, with a topology of N nodes, the traffic overhead is given by:

$$t = \sum_{i=1}^{N}\sum_{j=1}^{N} m_{i,j}$$

such as $m_{i,j}$ is the number of messages forwarded by node i and initiated by a node j.

Cache hit rate. This parameter, denoted by chr, is associated to DRDP approach. It defines the rate of messages that have been routed based on the cached information, without neither broadcasting the message to *all* neighbors nor a random walk forwarding. The chr value of $x\%$ means that out of 100 messages, including grant and search messages, x messages are routed based on the cache content.

D. Simulation results

Due to the high number of parameters, it was not possible to include all simulation results. In the remaining of this section, only a subset of these results is presented.

1) Baseline scenario: This simulation scenario used the simulation parameters reported in Table II. The simulation parameters have been selected to fit a plausible Grid scenarios. In particular, the wide range of resource classes, c_r, reflects the resource heterogeneity in such an environment. Moreover, the choice of the TTL value strongly depends on the network size. As we simulate a network of 10000 nodes, the ttl value was set to 3 ($ttl = 3$). A large TTL would result in covering almost all of the nodes when using the flooding scheme, which is unrealistic. Moreover, a low value of the TTL parameter may not provide a fair comparison of the random walk technique with both flooding and the DRDP approaches. For this reason, we kept the same configuration for the random walk approach and run additional simulations where the TTL value was set to 15. We assume for each generated query, a search peer requests n_{sr} distinct resources randomly selected such as $n_{sr} \in [1-5]$. As aforementioned, the search is considered as a success if and only if the n_{sr} resources are found. The replication ratio, p for all resources available in the network does not exceed 0.009 as shown in Figure 4.

Our objective through this scenario was to simulate the situation where a peer can not only request n distinct resources within a query but also requests scarce resources.

Figure 5 displays the success rate (s) during simulation time. Before conducting a deeper analysis of these results, we have to underline that the probability of a failed search increases with the number of requested resources in a single query. In that situation, a peer could succeed in locating a subset of the needed resources but not all of them.

As we can clearly see, at the beginning of the simulation period, DRDP and flooding schemes reach almost the same success rate. However, in steady state, DRDP achieves the highest success rate (around 0.57) and outperforms not only flooding (0.45) but also the random walks ($k = 15$, $ttl = 3$, $ttl = 15$) scheme in terms of successful searches. This can be explained by the fact that at the start phase, caches are almost empty (*cold cache phase*). So, as in the flooding approach, queries are sent to all neighbors and the TTL is decreased by 1 at each forwarding step (see Table I). Later, once peers build their per-link caches, and thus learn more information about the available resources in the network,

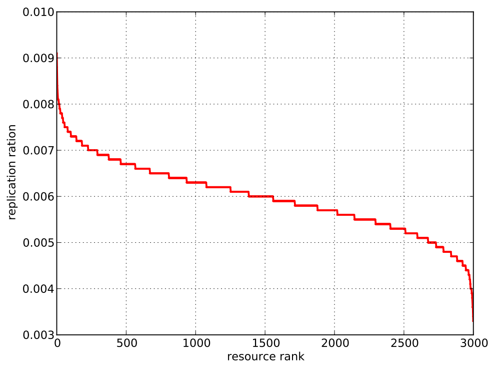


Figure 4. Resource distribution

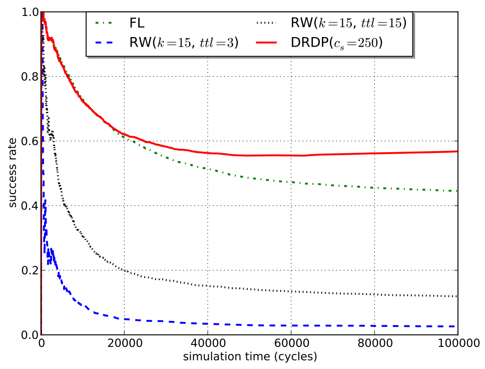


Figure 5. Success rate ($c_s = 250, ttl = 3$)

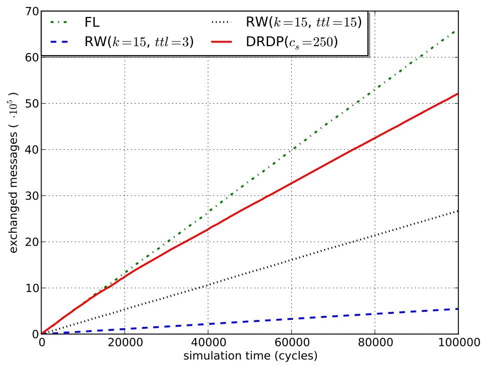


Figure 6. Traffic overhead ($c_s = 250, ttl = 3$)

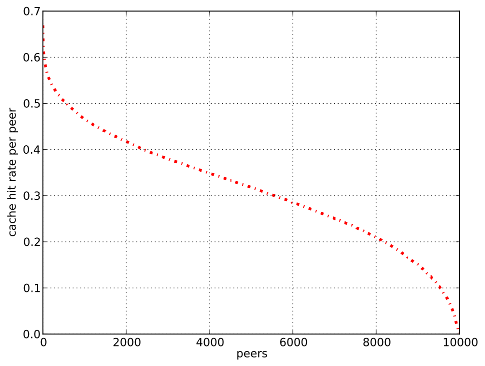


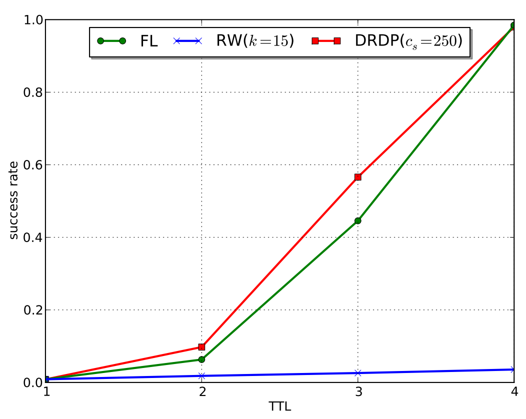
Figure 7. Cache hit rate distribution over peers ($c_s = 250$)

the probability to find a matching increases. Therefore, the dynamic TTL mechanism is used to efficiently forward the query farther than the initial query coverage given by the ttl parameter.

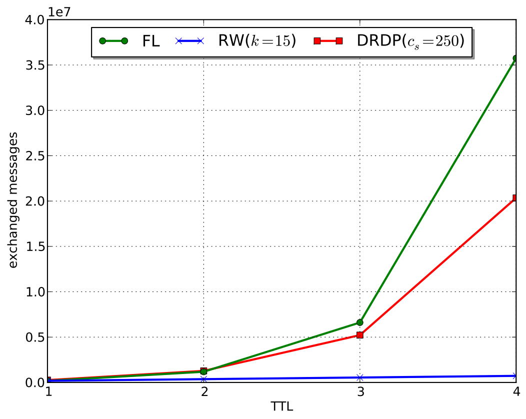
Keeping the same experimental settings, we measured the traffic overhead in terms of number of routed messages. We might expect DRDP to produce more overhead than flooding because of the grant messages. We find that this is compensated by the low number of request messages routed across the network. Therefore, as shown in Figure 6, the number of messages generated by DRDP is lower than that produced by the flooding. Again, due to the cache matching, the cache based forwarding quickly reduces the query flooding in favor of single forwarding. It is also clear that the random walks reduces considerably the traffic overhead but at the expense of a very low success rate. The distribution of the cache hit rate over peers is presented in Figure 7. It should be noticed that more than 50% of peers routed more than 30% of received messages based on their cached information. The low cache hit rate for some peers is because they are located outside of the message coverage and thus have an almost empty cache. This issue can be addressed either by increasing the initial TTL or by lowering the TTL decrement value (v), allowing messages to go farther.

2) Impact of TTL parameter: In this scenario, we evaluate the impact of message TTL on the performance of DRDP. We perform additional simulations investigating the behavior of the different schemes under various TTL values while keeping the same simulation parameters as in previous experiment.

Figure 8 depicts the success rate and the traffic overhead while varying the TTL values. As expected, both the success rate and the traffic overhead increase with the TTL. However, compared with flooding scheme, DRDP achieves the highest success rate and the lowest overhead when the TTL is respectively set to $1, 2, 3$. Although the initial success rate is similar when the TTL was set to 4, the traffic overhead generated by DRDP is approximately twice lower than the overhead produced by flooding. The main reason behind this behavior is that the query horizon increases with the TTL value. This favors the message duplication problem specially



(a) Success rate



(b) Traffic overhead

Figure 8. Success rate (a) and traffic overhead (b) under different ttl values

in blind search approaches.

Overall, the probability of finding a resource provided by the network strongly depends on the chosen TTL value: the wider the spreading of the query, the more matches are found and the more traffic is generated. Despite this tradeoff, DRDP approach still slightly affected, in terms of traffic overhead, while increasing the TTL.

3) Impact of cache size value c_s: In order to investigate the behavior of DRDP approach under several configurations, a set of simulations were performed to determine how the cache size influences the performance metrics defined above. As explained in Section III, the cache resource aggregation policy ensures a small cache size compared, for example, to the traditional cache approaches that typically store the resource providers or maintain result sets of resources previously discovered by *similar* queries.

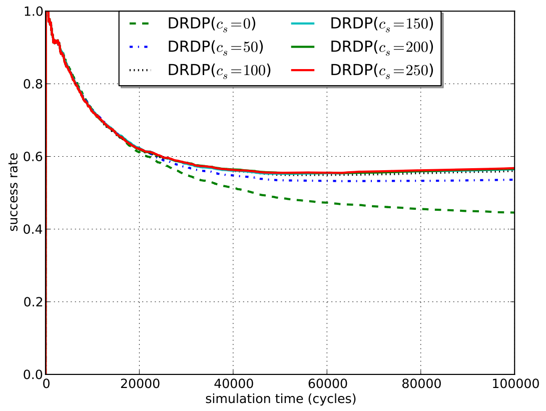
Let c_{smax} be the maximum cache size in terms of number of cached resources. Assume that, for each resource class, a peer p with a degree d receives at least one resource from each link. In that case, c_{smax} would be $c_{smax} = d \times r_c$. This means that, in all cases, c_s is upper bounded by $d \times r_c$, i.e., there is *at most* $d \times r_c$ cache entries in peer cache.

Consider the same simulation environment shown in Table II, Figure 9 exhibits the success rate (s) and the traffic overhead (t) while only varying cache size c_s. During the simulation time, and even with a small cache size $(c_s = 250)$, DRDP achieves high success rate and the lowest overhead. Note that when $c_s = 0$, DRDP behaves as flooding in the sense that we do not apply the dynamic TTL approach. As a result, the success rate achieved when $c_s = 0$ is the same as in the flooding approach (0.45) (see Figure 5). Increasing c_s can improve the success rate up to a given value n. Increasing it beyond this value will not have any impact neither the success rate nor on the traffic overhead. This can be clearly observed through Figure 10 which reports the variation of the success rate (s), the traffic overhead (t) and the cache hit rate (chr) when changing the cache size $(c_s \in \{0, 50, 100, 150, 200, 250\})$.

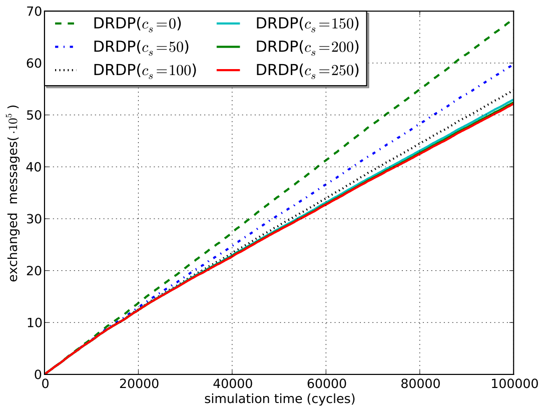
As we can see, the success rate (s) increases as the c_s increases before be stabilized around 0.57 when c_s was set to 250. Unlike s, the traffic overhead (t) decreases while increasing c_s. It is also maintained constant for $c_s >= 250$. It should also noticed that the cache hit rate (chr) increases with c_s and it is maintained around 0.35 (i.e., 35% of messages are routed based on the cache content) for the same value for c_s.

Note that under a P2P system when the resource replication ratio p is high, it is very common to receive the same resource description from all or a high number of neighbors. This can have an impact on the *peer resource view*, i.e., the number of distinct resources that the peer is aware of, specially when $c_s < c_{smax}$. Let us consider a simple scenario where we assume that a resource r has 2 entries in the cache, each of them associated to different peer cache link and new resource $r\prime$ is received when the peer cache is full. In that case, we believe that it is more efficient to remove an entry associated to r rather than removing another resource with only one entry. Roughly speaking, to avoid the occupation of one entry per link and so keep more global view about resources provided by the system, we may, for instance, specify a *threshold* value for the number of resource replicas that can be maintained in the cache. If the threshold value of resource r is reached, x replicas associated to r will be removed first.

4) Impact of DRDP TTL decrement value v: In order to study the impact of the DRDP decrement value (v), we have measured the success rate and the total number of messages routed within the network. Figure 11 demonstrates



(a) Success rate



(b) Traffic overhead

Figure 9. success rate (a) and exchanged messages (b) under different c_s values

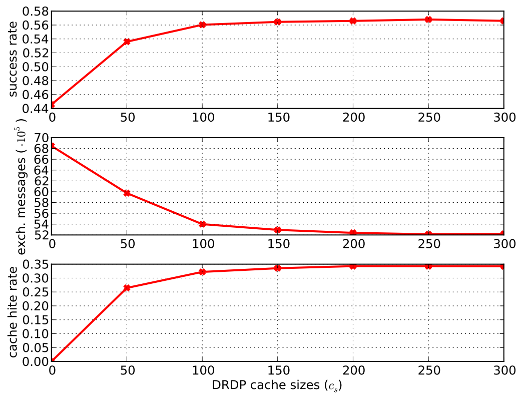


Figure 10. success rate (s), traffic overhead (t), cache hit rate (chr) under different cache sizes (c_s)

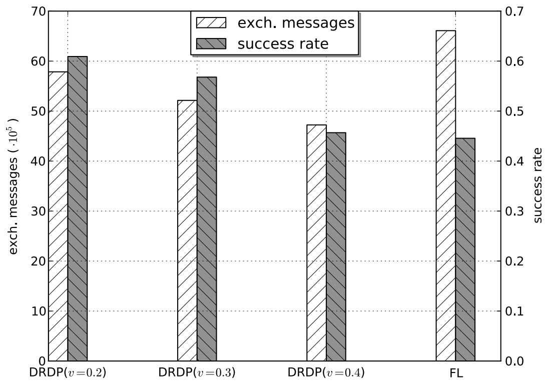


Figure 11. success rate (s), traffic overhead (t) under different TTL decrement values (v)

the performance of DRDP when varying the DRDP TTL decrement values (v) against the flooding approach. As we can see, comparing with a basic flooding, DRDP reaches the highest success rate at the lowest cost when v was respectively set to 0.2 and 0.3. Although the same success rate is achieved when $v = 0.4$, DRDP still produces the lowest overhead. We believe that further performance improvement can be achieved by adaptively tune the v value as function of *the number of required resources*.

V. Conclusion

We have proposed in this paper two schemes to improve the search in unstructured peer-to-peer networks. First, simple caches are associated to each link of a peer. These caches only store the resource description, not the owner, maintaining a small footprint and allowing aggregation. Second, a dynamic TTL scheme removes the horizon limitation of messages which are more likely to reach a peer providing the needed resource. They do not require a complex coordination between peers and can thus be added to existing informed search protocols to improve their performance.

To demonstrate the usefulness of these mechanisms, we have designed a simple search protocol, called DRDP, and investigated its performance through simulations reproducing a realistic application. Compared to flooding it yields to a lower number of messages despite the *grant* messages sent by peer holding resources. Compared to random walks, it greatly improves the success rate, especially with scarce resources or complex queries.

Although simple in design, DRDP proves complex to analyze due to the high number of parameters involved. Futher experiments may be carried out to evaluate the performance and scalability of our approach. This includes a dynamic network (i.e., peer churn) and adaptive tuning of DRDP parameters.

References

[1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2001, pp. 149–160.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content Addressable Network," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, vol. 31, no. 4. ACM Press, October 2001, pp. 161–172.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware)*, 2001, pp. 329–350.

[4] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing '02*. ACM, 2002, pp. 84–95.

[5] V. Dimakopoulos and E. Pitoura, "On the performance of flooding-based resource discovery," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 11, pp. 1242–1252, 2006.

[6] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks: algorithms and evaluation," *Perform. Eval.*, vol. 63, no. 3, pp. 241–263, 2006.

[7] "Gnutella RFC," http://rfc-gnutella.sourceforge.net/.

[8] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search (aps) for peer-to-peer networks," in *Proceedings of the Third IEEE International Conference on Peer-to-Peer Computin (P2P'03)*. IEEE Computer Society, 2003, p. 568292.

[9] F. Otto and S. Ouyang, "Improving Search in Unstructured P2P systems: Intelligent walks (i-walks)," in *Proceedings of the 7^{th} International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'06)*, ser. Lecture Notes in Computer Science, vol. 4224. Springer, sep 2006, pp. 1312–1319.

[10] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *In Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 5.

[11] H. Zhang, L. Zhang, X. Shan, and V. O. K. Li, "Probabilistic search in p2p networks with high node degree variation," in *ICC*, 2007, pp. 1710–1715.

[12] C. Gkantsidis and M. Mihail, "Hybrid search schemes in unstructured peer-to-peer networks," in *In Proceedings of IEEE INFOCOM*, 2005, pp. 1526–1537.

[13] R. Dorrigiv, A. Lopez-Ortiz, and P. Pralat, "Search Algorithms for Unstructured Peer-to-Peer Networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN '07)*. IEEE Computer Society, 2007, pp. 343–352.

[14] C. Doulkeridis, K. Norvag, and M. Vazirgiannis, "Schema caching for improved XML Query Processing in P2P systems," in *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 73–74.

[15] X. Luo, Z. Qin, J. Geng, and J. Luo, "IAC: Interest-aware caching for unstructured P2P," *International Conference on Semantics, Knowledge and Grid*, p. 58, 2006.

[16] D. A. Menascé and L. Kanchanapalli, "Probabilistic scalable p2p resource location services," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 2, pp. 48–58, 2002.

[17] S. Daswani and A. Fisk, "Gnutella UDP Extension for Scalable Searche(guess) v.01."

[18] "Gnutella2," http://www.gnutella2.com/gnutella2.

[19] L. Gatani, G. L. Re, and S. Gaglio, "An adaptive routing mechanism for efficient resource discovery in unstructured p2p networks," in *ICCSA (3)*, 2005, pp. 39–48.

[20] X. Bai, S. Liu, P. Zhang, and R. Kantola, "Icn: Interest-based clustering network," *Peer-to-Peer Computing, IEEE International Conference on*, vol. 0, pp. 219–226, 2004.

[21] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," http://peersim.sf.net.