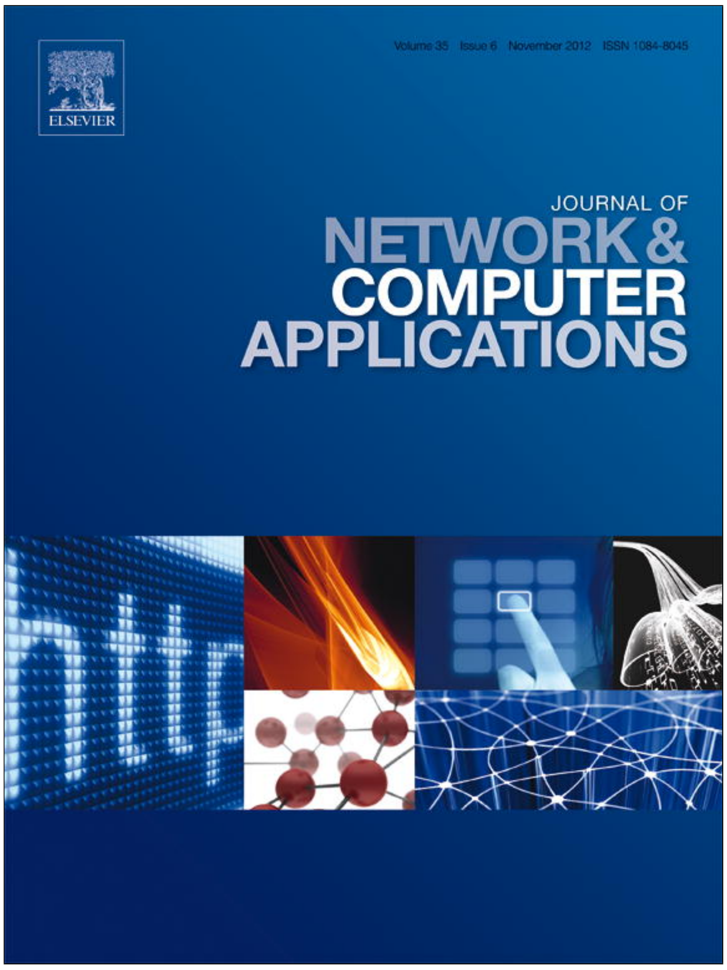

Volume 35 Issue 6 November 2012 ISSN 1084-8045
ELSEVIER
JOURNAL OF
NETWORK & COMPUTER APPLICATIONS

Journal of Network and Computer Applications 35 (2012) 1818–1830

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

HPM: A novel hierarchical Peer-to-Peer model for lookup acceleration with provision of physical proximity

Mourad Amad [a,*], Ahmed Meddahi [b], Djamil Aïssani [a], Zonghua Zhang [b]

[a] L.A.M.O.S., Laboratory of Modeling and Optimization of Systems, University of Bejaia, Algeria
[b] Institut Telecom/Telecom Lille 1, France

ARTICLE INFO

Article history:
Received 5 September 2011
Received in revised form
27 April 2012
Accepted 3 July 2012
Available online 13 July 2012

Keywords:
HPM
Hierarchical P2P
Routing optimization
Physical proximity

ABSTRACT

It is well known that Peer-to-Peer systems are generally featured with high flexibility and scalability, enabling dynamic resources localization and mutualization, and allowing the nodes to freely join and leave. But in some special environments such as mobile P2P networks, routing optimization, resources reliability and availability are critical concerns. To deal with these issues, we propose a novel architecture, termed hierarchical Peer-to-Peer model or HPM for short, based on Chord for improving P2P network performance in the presence of such additional requirements as fault tolerance and self organization. Specifically, HPM is composed of a set of hierarchical rings, each of which consists of the nodes that are both physically and logically close to each other or we say they have physical proximity, supporting inter and intra routing mechanisms. We show that the cost of lookup for HPM is $O(\sum_{i=1}^{4}\log_2(n_i))$, where n_i represents the number of nodes on ring level i (*with the maximum of 256 nodes in each ring in the case of four levels*). Each node maintains a routing table with only $2\times O(\log_2(n_i))$ entries, greatly facilitating HPM to work in resource-limited terminals such as mobile phones or PDA. In particular, when HPM is combined with a broadcast mechanism, the lookup process can be significantly improved (*four hops*). Our simulations and comparative studies demonstrate that HPM can achieve satisfactory P2P performance with rapid convergence thanks to the cost-optimal lookup algorithm.

1. Introduction

Peer-to-Peer (P2P) computing refers to a class of systems and applications that employ distributed resources to perform critical functions, such as resources localization in a decentralized manner. In other words, one of the challenges in P2P computing is to design a robust distributed system, which is composed of distributed and heterogeneous peer nodes, located in unrelated administrative domains (Balakrishnan et al., 2003).

According to different design goals, a number of system variants have been seen in P2P community. As defined in Peer-to-Peer Working Group (2001), P2P allows file sharing or computer resources and services by direct exchange between systems, or allows the use of devices on the Internet periphery in a nonclient capacity. In Steinmetz and Wehrle (2006), P2P is defined as a class of applications that take advantage of resources-storage, cycle, content, human presence-available in the Internet. Another salient feature is that P2P nodes must operate outside DNS systems, out the control of those central servers (Milojicic et al., 2002), since accessing to the decentralized resources implies unstable connectivity and unpredictable IP addresses in the operating environments.

The primary function of P2P networks is object location (*resource discovery and localization*) that means mapping an object ID to a node in the network, and retrieve this object with the same mapping (Aberer et al., 2005). For efficient routing, each node maintains $O(\log_2(n))$ pointers to other nodes in a typical P2P network, called neighbor pointers, where n is the number of network nodes. The number of hops required to locate an object at application layer in a typical P2P network is $O(\log_2(n))$ (Aberer et al., 2005). Each node stores neighbor pointers in a table (*finger table*). The design of protocols to construct and maintain consistent neighbor tables for network nodes that may join, leave, and fail concurrently and frequently is a key issue for distributed P2P networks.

The lookup cost constitutes also a critical issue for both structured and unstructured P2P networks. Nevertheless, it is reasonable to assume that the impact of structured P2P on the lookup cost complexity, topology maintenance for resources discovery and localization, in large scale networks, is more significant. This complexity and maintenance is even more critical and should be carefully considered when physical proximity is taken into consideration. Different from the structured P2P

* Corresponding author. Tel.: +213 551428098.
E-mail addresses: mourad.amad@univ-bejaia.dz (M. Amad), ahmed.meddahi@telecom-lille1.eu (A. Meddahi), lamos.bejaia@univ-bejaia.dz (D. Aïssani), zonghua.zhang@telecom-lille1.eu (Z. Zhang).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.07.006

network, the network topology of unstructured P2P networks is based on a random graph and flooding-based routing, and the flooding is only limited by a TTL (generating false negative).

In this paper, we propose a new structured Peer-to-Peer architecture for resources discovery and localization called *HPM*, it is derived from Chord. Nodes in HPM architecture are organized as a multilevels hierarchical set of rings, closed in terms of physical and logical proximities (*for simplicity, we describe the architecture in the case of four levels, and give a generalization for k levels*). Each level is composed of several rings. A ring is composed of at most 256 nodes. A node can belong simultaneously to two rings (*called relay node*). One of the main benefits of our proposed architecture is the rapid convergence with an optimized cost of the lookup process, while providing an efficient mechanism for fault tolerance and scalability. The size of the routing table (*finger table*) is also more compact compared to the main P2P architecture (e.g. *Chord*); thus making HPM well appropriate for terminals with limited or low capabilities such as PDA.

The paper is organized as follows: Section 2 gives a brief overview of P2P networks, with a focus on the lookup problem. The key elements of structured P2P systems such as distributed hash table (DHT) are described. Related works on hierarchical P2P networks are detailed on the end of section. Section 3 presents and describes the proposed HPM architecture. Performance evaluation is given in Section 5. Finally, we conclude and give some perspectives, especially related to security aspects.

2. Background and related work

Peer-to-Peer is relatively new in the areas of networking and distributed systems and services (e.g. *VoIP*) (Singh and Schulzrinne, 2004). P2P systems are characterized by several generations, with transitions between generations motivated by different goals. In this section, we describe and analyze the different generations of P2P networks through some illustrations.

2.1. Chronological apparition of P2P Networks

The first P2P generation started with Napster files sharing application. The main contribution of Napster was the introduction of a network architecture, where machines are not categorized as client and server, but rather as machines that offer and consume resources (*Servant*). All participants have more or less the same functionality. However, in order to locate files in a shared space, Napster provides a central directory. It is composed of two services: a decentralized storage service, but with a centralized directory service that constitutes a single point of failure, and makes it sensitive to denial of service attack.

The single point of failure due to the central coordination in the first solution (*Napster*) leads to the transition of a new kind of P2P systems, aims to eliminate the central coordination. The second generation of P2P systems started with Gnutella application solving the problem of the central coordination, using a purely distributed architecture, based on a flooding technique for lookup. However, the problem of scalability constitutes a critical issue in large network, due to the network traffic load generated by the flooding mechanism for research and localization. Moreover, Gnutella system does not guarantee to locate an existing data item (*false negative*), This is essentially due to the limited search scope (*TTL, generally equal to seven such as in http*).

To reduce network traffic, clustering based solutions have been proposed. These solutions use generally a super nodes (Xu, 2005; Bai et al., 2004; Tong et al., 2005; Chao et al., 2006; Li and Vuong, 2004; Miasnikov et al., 2004; Lee et al., 2007; Joung and Lin, 2002) (*super nodes or super peers are nodes with higher capabilities; they have additional functionalities compared to the ordinary nodes*). However, clustering based solutions present some disadvantages. First, the search mechanism (*lookup*) is not deterministic as it is based on flooding mechanism with a predefined scope. Second, the choice of super peers constitutes a key element for such systems. If the super peers are static, they can be exposed to denial of service attacks (DoS), and in case of dynamic super-peers, a significant overhead is needed for system stabilization (*self-organization*), when these super peers join or leave the system.

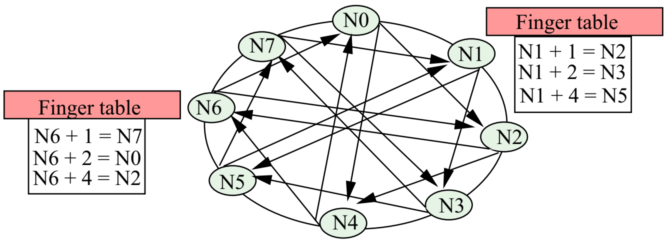


Fig. 1. Chord architecture.

The third generation of P2P systems initiated by research projects such as Chord (Stoica et al., 2003), CAN (Ratnasamy et al., 2001), Tapestry (Zhao et al., 2004a), Pastry (Rowstron and Druschel, 2001), *Chord*^{2} (Joung and Wang, 2007), Cycloid (Shena et al., 2006) and Kademlia (Maymounkov and Maziére, 2002) use distributed hash table (DHT) to generate a key for both nodes and data. A node (*peer*) in such system requires a unique identifier, based on a cryptographic hash of some unique attribute such as its IP address. Node identifier and key value pairs are both hashed to one identifier space. The nodes are then connected to each other in a certain predefined topology (e.g. *circular space* (*Chord*), *td-dimensional cartesian space* (*CAN*)). Figure 1 shows the Chord architecture with 8 nodes.

Structured P2P networks (*third generation*) present three main drawbacks, First, DHT is designed for exact-match query thus limiting keyword searches. Second, substantial repair operations are required in case of high churn rate. Third, hot-spots are generated for too frequently accessed files (Shena et al., 2006). However, the cost of lookup is much more optimized, greedy and deterministic. So if a resource exists in the system, the requester node is able to locate it with a minimal cost of lookup (*in terms of number of hops*).

The common objective in all generation of P2P systems is to optimize resources discovery and localization or lookup (*in terms of number of hops*), but also the overhead, especially in a dynamic and heterogeneous P2P system. Lookup problem can be resumed as follows: a publisher insert an item *X* (e.g. *a file or resource*) in a dynamic system, while somewhere else a consumer may access and retrieve *X*. More generally, when the consumer is connected to the system, how does the resource is located?

Since key lookup is probably the most frequently executed operation, and essentially on all DHT systems (*third generation*), a focus is done on lookup process performance. One of the main performance criteria is the number of routing hops (*cost of lookup*) which is a key factor for end-to-end latency. Nevertheless, latency for each hop is relative to physical proximity, and plays also an important role. Generally, adding some extra routing information on each node increases the probability for providing better routes. However, information and links management in the system generate overhead in terms of processing time and bandwidth consumption.

Because DHT is considered as a key element for the third generation of P2P systems; but also for our proposed HPM, we give a brief overview of the functional principle of DHT in the next sub-section.

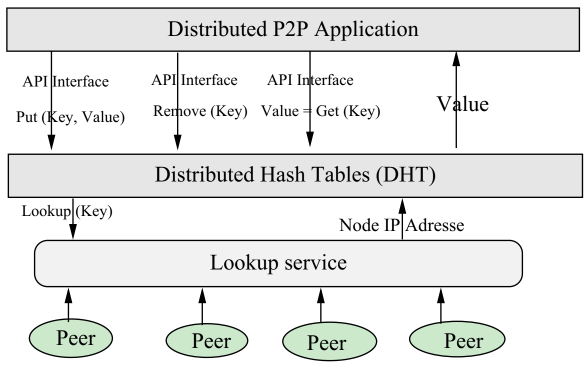


Fig. 2. The DHT functional principle (Lua et al., 2004).

2.2. Distributed hash table (DHT)

A hash-table interface is an attractive foundation for a distributed lookup algorithm, because it places a few constraints on the keys structure, but also on their associated resources. Hash-table maps efficiently "keys" into "values". The main requirement is that resource (*or data*) and nodes storing keys for each other (*key responsibility*), can be identified using unique numeric key. This organization is different from Napster and Gnutella, which search for key words. The DHT implements one main operation: `Lookup (Key)` resulting in a node identity (e.g. *IP address and port number*), currently responsible for the given key. A simple distributed storage application could use the interface as follows (see Fig. 2): a particular and unique name is used to publish a file or resource, it is converted to a numeric key using an ordinary hash function such as SHA-1 or SHA-2 (`Get (key)`), and then a `lookup (Key)` function is called. The publisher sends the file or resource (`Put(Key, Value)`) to be stored at the resulting node (*data replication*). Then the requestor converts the file identifier to a key, calls the `Lookup (Key)` function, and requests the resulting node for a file or resource copy.

Typically, DHT has the following design constraints:

- *Few neighbors*: each node should maintain only a small number of active neighbors, storing for each neighbor its physical IP address (*from a P2P perspectives*), and its logical key partition. A typical constraint for the number of neighbors is $\log_2(n)$ for a network with n machines (*peers*). By keeping this number small, the arrival or departure of nodes from the network, results in a limited and bounded number of update messages $O(\log_2(n))$ in a typical DHT.
- *Low latency*: each node should be reachable from any other node, with a limited number of network routing hops; this requires that the neighborhood graph gets a small diameter. Again, $O(\log_2(n))$ is a typical constraint.
- *Greedy routing decisions*: each node should be able to decide how to forward keyed messages without consulting any other node. These greedy decisions must find the short (*typically* $O(\log_2(n))$) path between any pair of nodes.
- *Robustness*: as nodes and links join and leave; the network should remain mostly connected, and able to route packets. A necessary condition for these features is a network with a nontrivial min-cut, to allow multiple alternate routes for packets.

Figure 2 shows a typical DHT architecture.

2.3. Hierarchical P2P systems

Hierarchical P2P systems have been a research topic since the introduction of P2P systems. They make sense to organize the network in two or more hierarchical levels, while enabling distinction between nodes with different capabilities. Efficient nodes, with higher lifetime, belong to the highest hierarchical level, while low performance nodes participate in the lower levels. Thus, the impact of their short online times on the system operation is reduced as much as possible. Membership dynamicity is considered to be a critical issue for DHTs based mechanisms. The capability to better fit to the physical network is cited in the literature as a further advantage of hierarchical systems (Zoels et al., 2008).

In hierarchical DHTs, peers are organized into groups, each group has its autonomous intra group overlay network and lookup service. To find a peer that is responsible for a key, the top level overlay first determines the group responsible for the key, the responsible group then uses its intra group overlay to determine the specific peer that is responsible for the key. However, in our proposed HPM, the top level is not covered by all requests, but only by request where the searching key is on this level. Consequently, the request load balancing is guaranteed.

Garces-Erice et al. (2003) describe a hierarchical architecture based on a Chord overlay network that can be used to improve the routing performance. Super-peers save the information of all their leaves while peers just send keep-alives messages to their super-peer. This implies that super-peers are getting more stressful if the number of peers increases. In Zoels et al. (2006), an analysis of the costs of super-peers on a hierarchical structure is done. Peers in each cluster do not maintain any structure and rely only on their super-peer. The authors of Martinez-Yelmo et al. (2008) propose a hierarchical architecture based on super-peers, where a peer ID is composed by a Prefix ID and a Suffix ID. Prefix ID is only routed at the super-peer level and the Suffix ID at the peer level. Jelly (Hsiao and Wang, 2004) uses a node joining mechanism as a fine-tuning tool similar to Grapes (Shin et al., 2002) and a dynamic checking mechanism as a coarse-tuning tool to balance the hierarchy.

Our proposed HPM architecture (*HPM: a novel hierarchical Peer-to-Peer model for lookup acceleration with provision of physical proximity*) is derived from Chord and based on hierarchical rings. HPM takes into consideration the physical proximity and neighborhood, with no distinction between nodes (*peers*) on different levels. In HPM, the physical proximity of nodes belonging to level $i+1$ is higher than that of nodes at level *i*. So, the physical proximity increases as the ring level increases.

The process of routing in P2P networks (*lookup data*) operates at application layer. The overlay P2P network may lead to routing inefficiency, as opposed to routing service provided by the transport layer (IP). Our proposed HPM routing objectives are: (1) minimizing the number of hops and delay, (2) locating nodes that store data or resources in purely decentralized P2P networks, and (3) but also, controlling the overhead while considering the physical proximity of nodes.

In this context, HPM is considered as a scalable P2P protocol that optimizes resources discovery and localization function in a decentralized manner. It is based on cryptographic hash function for resource identifier, IP addresses and port number for node identifier. In existing DHT approaches, node and resource identifiers are obtained from name or meta-information using SHA-1 algorithm. However, in HPM approach, node identifiers are simply obtained from IP address of nodes. This is very important for four mains reasons: (1) HPM node identifiers can be rapidly generated from their IP address compared to SHA-1 algorithm using in existing DHTs, (2) except in some very few cases such as NAT traversal, IP addresses are supposed to be unique, so IDs collision probability is extremely low, (3) physical proximity is taken into account in our approach HPM, as most of the nodes located in the same IP network domain; belong to the same HPM ring or to rings that are close to, and (4) HPM architecture is organized in such a way that the majority of nodes are placed on level 4 (256^3 *rings*) and are mainly responsible for storing resources (*less for routing*), while the rest of the nodes are placed on the lower levels which are mainly responsible for routing process (*less for storing*). This organization leads to a better load balancing between nodes, for routing and storing.

HPM architecture belongs to the third generation of P2P systems (*structured architecture*), which is based on specific topology that consider physical proximity.

Table 1 gives a synthetic comparison between HPM and some other hierarchical DHTs. HPM does not use any existing architecture as in Grapes or Jelly, but it uses its proper architecture. HPM takes a better consideration of the physical proximity than the main existing hierarchical DHTs.

The following section describes and analysis the HPM architecture.

3. HPM: concept, principle and architecture

HPM is organized as a set of hierarchical rings based on a multilayers topology (see Fig. 3). The IP address is splitted into four equal parts such as p1, p2, p3, and p4, where $0 \leq pi < 256$. Then, each part (pi) can take 256 values while each ring is composed of at most 256 nodes. Each layer or level i is composed of 256^{i-1} rings that connects neighboring nodes as follows:

- On level 1 ($i=1$), there is one ring with a maximum of 256 nodes, $256^{(1-1)} = 256^0 = 1$ ring;
- From each node on ring level 1, a ring will be constructed on level 2 ($i=2$), then 256^1 rings will be constructed ($256^{(2-1)} = 256^1$);
- From each node on rings level 2, a ring will be constructed on level 3 ($i=3$), then 256^2 rings will be constructed ($256^{(3-1)} = 256^2$);

Table 1
HPM vs some other hierarchical DHTs.

Scheme	Architecture	Routing	Physical proximity
Grapes (Shin et al., 2002)	Each sub-group of nodes are managed by a super peer. The super peers form the second level	Flooding mechanism	Yes, but only on sub-groups
Jelly (Hsiao and Wang, 2004)	Such as in Grapes, but the two levels are organized on existing DHT	Existing lookup DHT	Only on sub-groups
Hierarchical Kademlia (Martinez-Yelmo et al., 2008)	Hierarchical DHT, prefix node IDs is used for connecting super peers and suffix IDs for connecting nodes in the same clusters (*Kademlia*)	Based on the prefix and sufix	No physical proximity consideration
Hierarchical Chord (Garces-Erice et al., 2003)	Chord with two layers	The first level for routing and the second one for maintenance	No physical proximity consideration
HPM	Hierarchical rings with no explicit super peers	Deterministic and greedy	Yes, physical proximity increases as the ring level increases

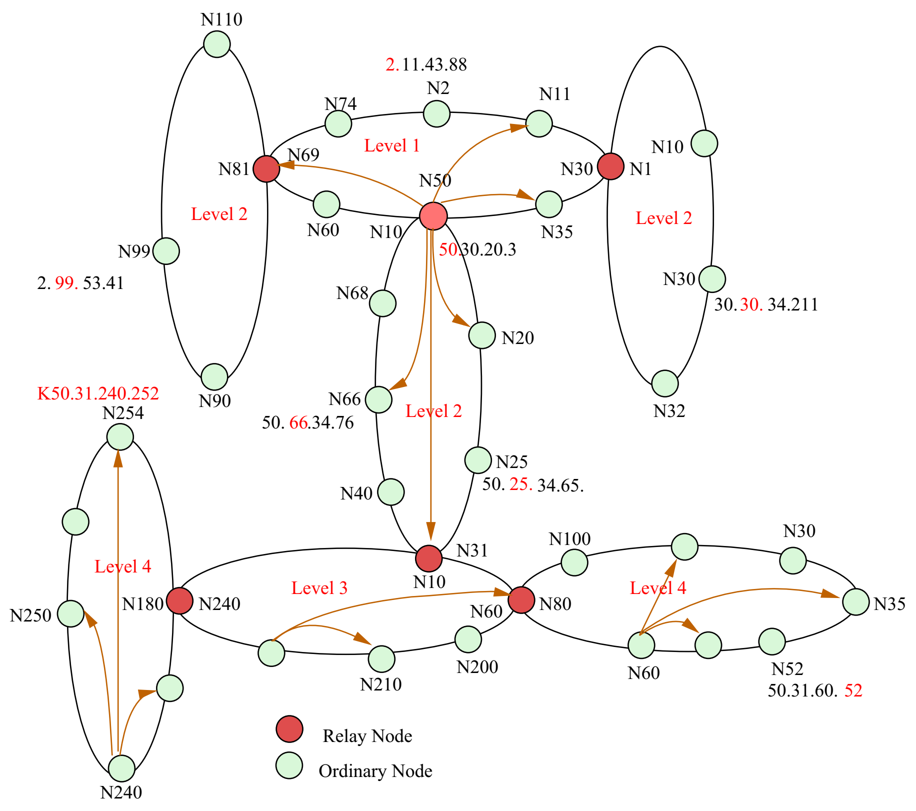


Fig. 3. HPM architecture.

- From each node on rings level 3, a ring will be constructed on level 4 (i=4), then 256^3 rings will be constructed ($256^{(4-1)} = 256^3$).

For simplicity and illustration purposes, we describe the architecture in case of four levels and then, we generalize the topology for k levels.

In HPM, each node n is identified by a unique identifier, which is the ith part of its IP address divided on four equal parts ($1 \le i \le 4$, *in case of four levels*). i represents the level to which the node n belongs. The resources are also identified by a unique identifier, generated by the distributed hash tables using some cryptographic hash functions (*for load balancing*). Each resource key is also composed of four parts (a, b, c, and d). From the example illustrated in Fig. 3, the node with IP address: **2**.11.43.88 on ring level 1, gets the identifier N2 (*first part of its IP address*). The identifier of IP node: 50.**66**.34.76 on ring level 2 is N66 (*second part of its IP address*); this node does not belong to ring level one, as a node with identifier N50 already exists on the first level. The node with IP address: 50.31.60.**52** on level 4 gets the identifier N52, as there are already node N50 on level 1, node N31 on level 2, and node N60 on level 3.

As an example, when a new node $n1$ with IP address 125.11.23.107 join the HPM system, if there is no node with identifier $N125$ on level 1 (*node with IP address 125.x.y.z*), the node $n1$ will be placed on level 1 with identifier $N125$ (*first part of its IP address*).

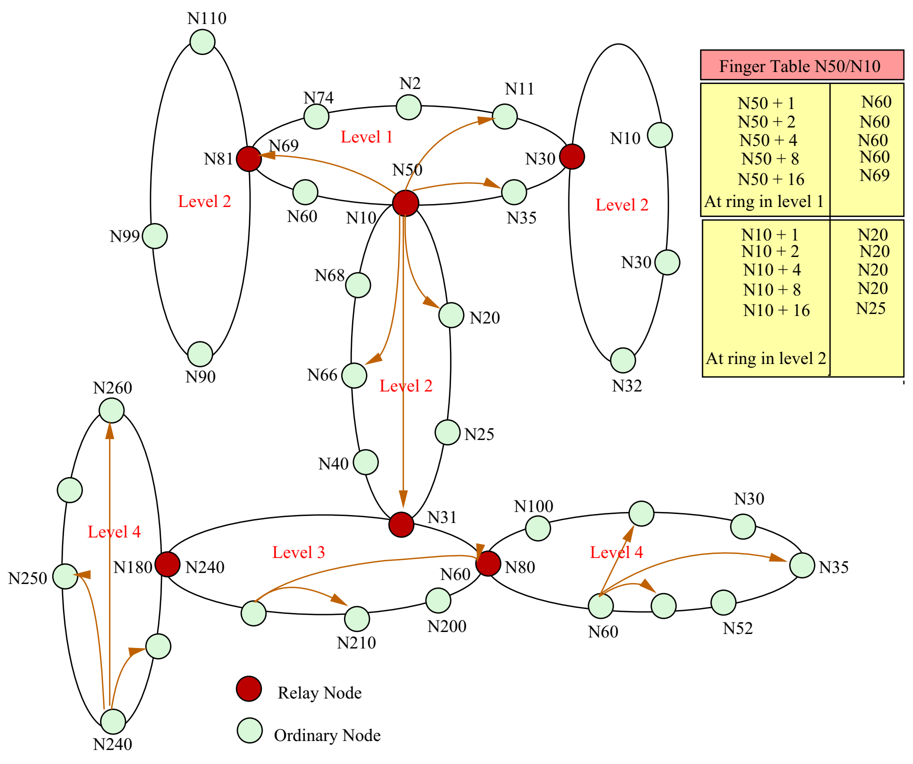
When another new node $n2$ with IP address 125.12.59.77 join the HPM system, it will not be placed on level 1 with identifier N125 (first part of its IP address), because there is already a node n1 with this identifier. Then $n2$ will be placed on level 2 with identifier $N12$ (*second part of its IP address*).

When another new node $n3$ with IP address 125.12.34.88 joins the HPM system, it will not be placed on level 1 with identifier $N125$ (*first part of its IP address*), because there is already a node $n1$ with this identifier. It will not be placed on level 2 with identifier N12 (*second part of its IP address*), because there is node $n2$ with this identifier, then $n3$ will be placed on level 3 with identifier $N34$ (*third part of its IP address*).

When another new node $n4$ with IP address 125.12.34.54 join the HPM system, it will not be placed on level 1 with identifier $N125$ (*first part of its IP address*), because there is node $n1$ with this identifier. It will not be placed on level 2 with identifier $N12$ (*second part of its IP address*), because there is node $n2$ with this identifier, it will not be placed on level 3 with identifier $N34$ (*third part of its IP address*), because there is node $n3$ with this identifier. Then, node $n4$ will be placed on level 4 with identifier N54 (*fourth part of its IP address*).

The main characteristic of the proposed architecture is the rapid convergence of the lookup process, with a limited overhead. HPM approach allows each node to maintain minimal state information. Each node maintains only $2 \times O(\log_2(n_i))$, where n_i is the number of nodes on one ring, and in case of four levels, $n_i \le 256$. Thus, HPM is well adapted for terminals with limited resources and capabilities, such as mobile devices (e.g. *PDA and mobile phone*).

The HPM architecture is based on structured and hierarchical rings. Each ring has 256 (2^8) nodes (*maximum*). The first level is composed of one ring (*super ring*) and contains the nodes with IP addresses that are different in the first part, with no restriction on the other parts. It is recommended but not necessary that these nodes are stable nodes such as in classical hierarchical P2P architectures. From the example illustrated in Fig. 3, a first node n with IP address: 176.*x.y.z*, belongs to level 1, and gets identifier N176, while the other nodes with IP address, such as 176.*a.b.c* do not belong to the same ring (*level 1*) as node n, but to other sub-rings at lower levels (*2, 3 or 4*). Each level connects a maximum of 256^{i-1} rings (*i corresponds to the number of level*). One of the main HPM characteristics is that nodes on the same network domain with IP address such as: 50.31.60.**123**, 50.31.60.**125**, 50.31.60.**150** and 50.31.60.**209** belong to the same ring in the same level (*level 4*). In this way, the physical and logical proximities are somehow taken into account. On each ring of each level, nodes are organized and ordered increasingly based on their identifiers. Each node maintains



Finger Table N50/N10	
N50 + 1	N60
N50 + 2	N60
N50 + 4	N60
N50 + 8	N60
N50 + 16	N69
At ring in level 1	
N10 + 1	N20
N10 + 2	N20
N10 + 4	N20
N10 + 8	N20
N10 + 16	N25
At ring in level 2	

Fig. 4. Finger table in HPM.

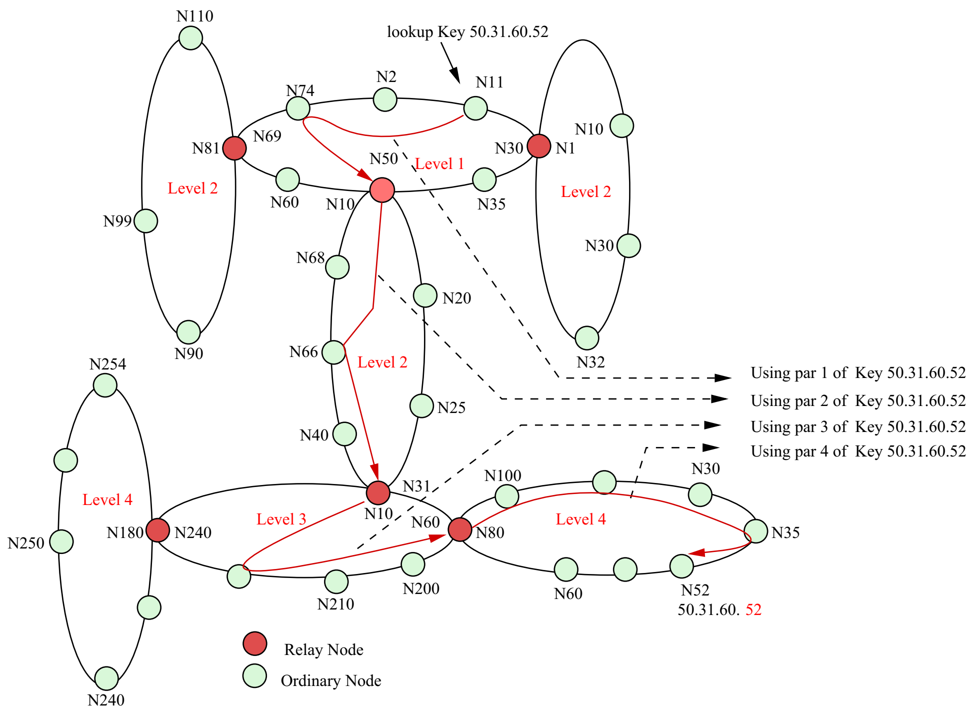


Fig. 5. Example of lookup in HPM.

a routing table with IP addresses and port numbers, corresponding to the neighboring nodes. Nodes in private network are managed and represented by their server (NAT) (see *RFC 1918 and RFC 4193*). Figure 3 illustrates the HPM architecture.

Each resource with an "*a.b.c.d*" type identifier, will be placed and located at the node with IP address *w.x.y.z*, where *w* (*respectively, x, y, z*) is the lowest value greater or equal to *a* (*respectively, b, c, d*). In Fig. 3, data with key K50.31.240.252 is placed on node N254, with IP address 50.31.240.254 on level 4, because 50 (*respectively, 31, 240 and 252*) is the lowest value greater or equal to 50 (*respectively, 31, 240 and 254*).

Most of the nodes in HPM architecture are placed on rings at level 4 (see Section 4). In this level, nodes in each ring are logically close, their IP addresses are similar in the first three parts, and they are different only in the fourth part. This means that they are in the same segment (*physically close*).

3.1. Finger table in HPM

Let be m the number of bits in the space of node identifiers on one ring (e.g. $m=8$ *for 256 nodes*). Each node n maintains a routing table of at most m entries, called the `finger table`. The ith entry in the finger table of node n contains the identifier of the first node s, that succeeds n by at least 2^{i-1} on the identifiers circle, where $1 \le i \le m$. We call node s the ith finger of node n. A finger table entry includes both HPM identifier, IP address and port number of the relevant node. Node participating on one ring has a similar finger table as in Chord, but if this node participates on two rings, it gets a double finger table entries (*one for each ring*). Figure 4 shows the finger table of a node with IP address **50.10**.20.3, belonging to two levels (*dark node, called also relay node*), and then has two identifiers: N50 on level 1 and N10 on level 2. This node participates on these two rings for lookup and stabilization with the same manner.

3.2. Lookup process in HPM

For each ring k at level i, we use the ith part of the data key for the lookup process, as in Chord. If the request succeeds on this ring k, where the requestor node belongs to, the cost of lookup is $O(\log_2(n_i))$ (n_i *is the number of nodes on this ring,* $n_i \le 256$). In case where the resource does not exist on the active covered ring, the search or localization is done on ring level $i+1$ or $i-1$, in a deterministic manner (*greedy routing*), then the cost of the lookup process is $O(\sum_{i=1}^{4} \log_2(n_i))$, where n_i is the number of nodes on the covered ring at level i (*on which the request succeeded*).

Figure 5 shows an example of the lookup process, node $N11$ search key $K50.31.60.52$, it uses key $k50$ on the first level based on the finger table as in Chord, with the same manner $K31$ on the second level, $K60$ on the third level and finally $K52$ on the fourth level. The key will be located on node $N52$. Algorithm 1 gives a pseudo code for the lookup process in HPM.

Algorithm 1. Lookup data pseudo code in HPM.

```
Lookup (Key $c_1c_2c_3c_4$)
1: Begin
2: Locate the node $X$ (in the same ring) of IP Address $p_1p_2p_3p_4$,
   where $p_i$ is the smallest value greater or equal to $c_i$,
   $\forall i \in [1 .. 4]$.
3:    If $\exists c_j$ where $c_j > p_j$ and $j < i$ Then
4:    go to level $(i-1)$ and call fig15 (Key $c_1c_2c_3c_4$) (if $i > 1$,
      otherwise, data does not exist)
5:    Else
6:       If the data is present Then
7:       Loading data from the resulting node.
8:       Else
9:       go to level $(i+1)$ and call Lookup (Key $c_1c_2c_3c_4$) (if $i < 4$,
         otherwise, data does not exist)
10: End.
```

Theorem 1. *The number of hops between any two arbitrary nodes in HPM using the greedy Algorithm 1 is* $\sum_{i=1}^{4} \log_2(n_i)$, *where* n_i *is the number of nodes in each covered ring.*

Proof. In one ring, each node maintains a routing table of m entries; the ith entry contains the identifier of the first node s that succeeds n by at least 2^{i-1}, then the number of hops between any two nodes in one ring is $\log_2(n_i)$, where n_i is the number of nodes in this ring. For the HPM lookup process, only one ring is covered at each level, so to cover the four levels, the cost of lookup is $\sum_{i=1}^{4} \log_2(n_i)$. □

3.3. *Maintenance, stabilization and fault tolerance*

Maintenance, stabilization and fault tolerance of HPM architecture is efficiently ensured on one ring as in Chord, and between rings by two additive links as follows: the predecessor of each node belonging to two levels (*relay node*) is connected to its successor in another ring. Thus, when this node fails, the network is kept globally connected. As shown in Fig. 6, node N69/N81 (*belongs to levels 1 and 2*) has two immediate predecessors, N60 at level 1 and N110 at level 2, which are connected, respectively, to their two immediate successors N90 at level 2 and N74 at level 1.

3.3.1. *Node join*

Another important issues is to maintain an active topology and preserve the ability to locate and update each key in the network. For this, the bootstrapping constitutes a vital core functionality, required by every Peer-to-Peer overlay network. Nodes intending to participate in such overlay network; initially have to find at least one node that is already part of this network. Like in Cramer et al. (2004), four solutions applicable for the bootstrapping problem exist, and are resumed as follows:

- *Static overlay nodes-bootstrapping servers*: in original P2P network like Gnutella, initial bootstrapping was solved by placing static nodes (e.g. *enrolment servers*) in the overlay. This bootstrapping method requires low complexity, but at the cost of scalability.
- *Out-of-band address caches*: in P2P overlay, nodes actively report suitable nodes to HTTP-based caches. Nodes joining the system contact the caches that are accessible via URLs, in order to get a list of IP addresses.
- *Random address probing*: for large-scale overlay networks, the random address probing could be adapted for bootstrapping. A node entering the overlay network; randomly generates an IP address from the global address space, and then tries to establish a transport connection to this IP address using a well-known port. In case of connection failure, another address has to be defined and tested.
- *Employing network layer mechanism*: the discovery of overlay nodes process during bootstrapping should be based on the topological structure of the underlying network. If overlay nodes exist in the same network segment, it is convenient and highly efficient for network layer mechanisms to connect theses nodes, at least for bootstrapping. In a multicast capable network, a multicast group can be established for bootstrapping purposes.

To reduce system complexity, static overlay nodes-bootstrapping servers (see Fig. 7) is considered for our proposed HPM architecture.

To participate in the network, two steps are needed for the new node. After a join operation, the node initializes its finger table (*step 1*), and gets a part of the resource key from its neighboring nodes (*step 2*: *transfer of key responsibility*). The description of these two steps is detailed below.

Finger table initialization: using nodes given by the bootstrap server, the joining node will be placed in the network, and then initializes its finger table by exchanging information with its neighboring nodes. Remember that the ith entry in the finger table at node n contains the identity of the first node s that succeeds n, by at least 2^{i-1} on the identifier circle. If node n belongs

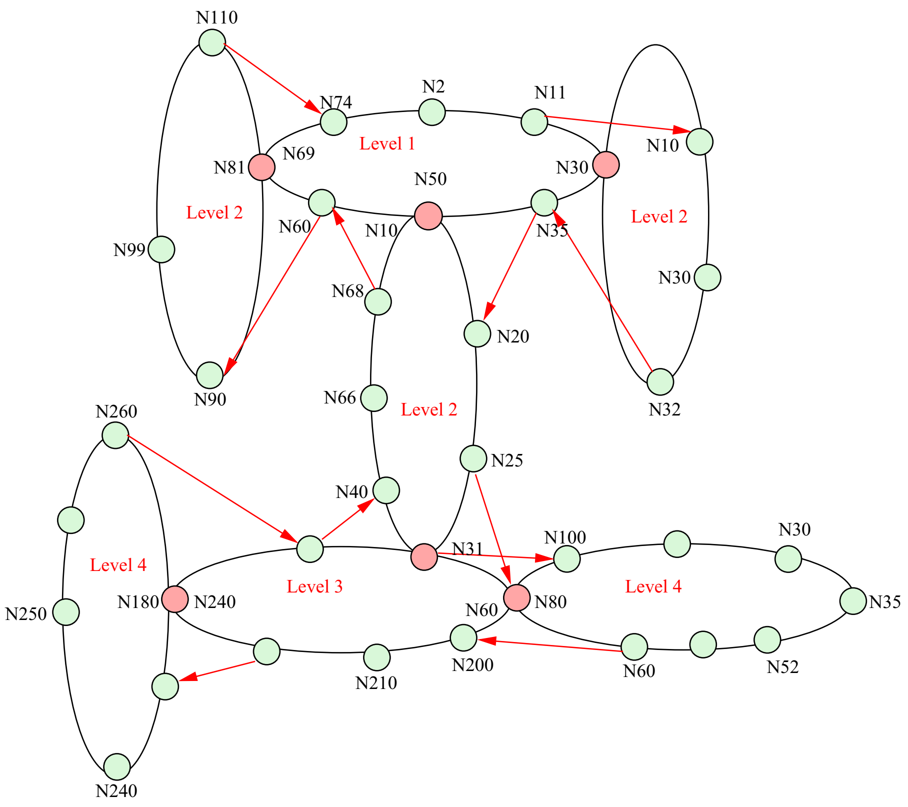


Fig. 6. Additive links in HPM for maintenance and stabilization.

simultaneously to two levels (*it will has two identifiers*), its finger table contains two similar parts, one for each level.

Transferring keys: when a node *n* joins the HPM system, there is a transfer of key responsibility to this new node *n*. All keys (*for which node n is now the new immediate successor in the same ring*) are transferred from the previous successor *s*. The node *n* becomes the successor only for keys that where previously managed by the successor *s*, and (*keys* $\in$ *]p, n], where p is now the predecessor of n*). So, *n* needs only to contact this unique node *s* to get all relevant keys.

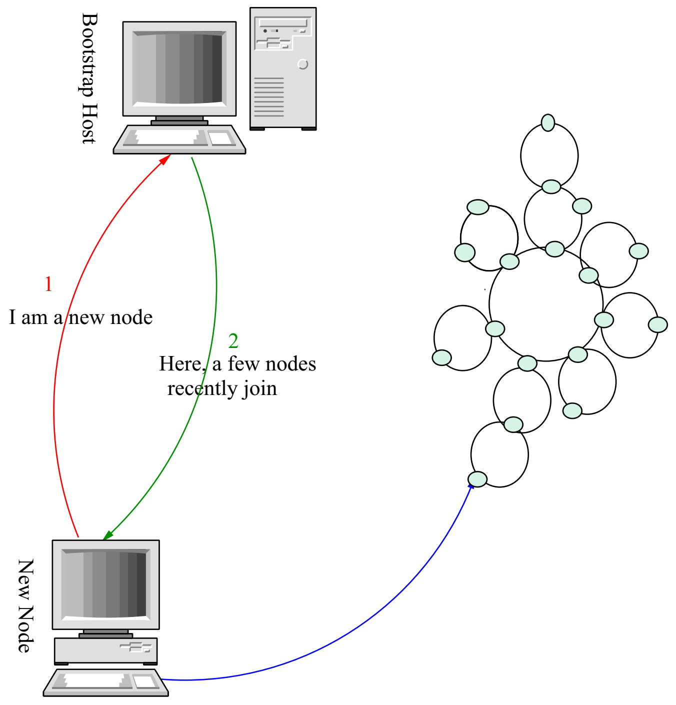


Fig. 7. Bootstrapping process used in HPM.

3.3.2. *Node leave*

When a node *n* belonging to one level leaves the system, the nodes (*successors and predecessors which have a pointer to n*) must update their finger tables, following some transfer key operations. In case, where the leaving node belongs to two levels (*relay node*), the stabilization algorithm (Algorithm 2) is activated at each neighbor node of *n*. For this algorithm, we use the following notations: *n*: the identifier of the failed node (*leaving node*), and l_i, l_{i+1}: the two levels to which node *n* belongs.

Algorithm 2. Stabilization pseudo code in HPM.

```
leave $(n, l_i, l_{i+1})$
1: Begin
2:   If $(l_{i+1}$ is null) Then // n is not a relay node
3:   Update the routing tables
4:   Else // n is a relay node and belongs to two levels
5:     If $\exists(n', l'_i, l'_{i+1})$ where $l'_i = l_{i+1}$ Then
6:       If $(l'_{i+1} = \text{null})$ Then
7:       $l'_i = l_i;\ l'_{i+1} = l_{i+1};$
8:       Update the routing tables
9:       Else
10:        $l'_i = l_i;\ l'_{i+1} = l_{i+1};$
11:        update the routing tables
12:        leave $(n', l_i, l_{i+1})$
13: End
```

The lookup service in HPM system is a continuous process (*always available*), even in case of several node failures, as each node has

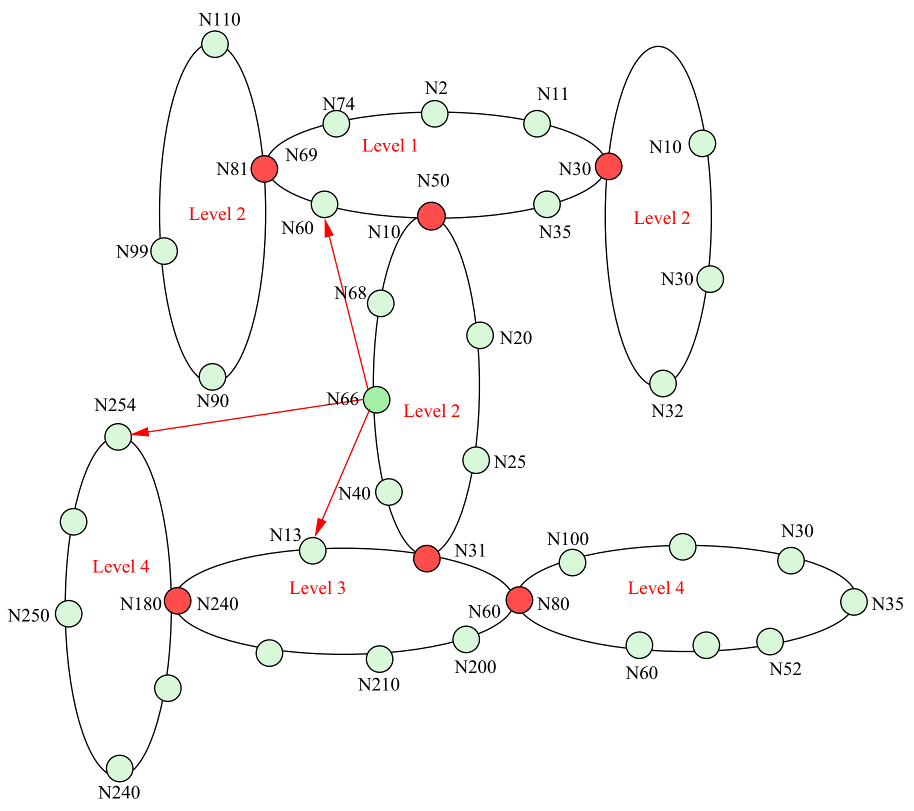


Fig. 8. Additional links in HPM for lookup acceleration.

$O(\log_2(n_i))$ successors ($n_i \le 256$), and then $(\log_2(n_i))^4$ possible routes from the "key" requestor node to the "key" locator node.

3.4. Lookup acceleration in HPM

The basic HPM lookup process as described in Section 3.2 can be accelerated for specifics applications or conditions; when it is combined with some known mechanisms, such as additional links or broadcast techniques.

Acceleration using additionally links: for accelerating the lookup process, each node n in any ring on level j maintains $l-1$ (*three*) additional links as shown in Fig. 8. Each additional link connects a node n to another node in ring level i, where $1 \le i \le 4$, and $i \ne j$. As an example, the three additional links for node N66 on level 2 are: N60 on level 1, N13 on level 3 and N254 on level 4.

A node N_j searching for a key (*lookup*) sends simultaneously a request `Lookup-AL (Key c1c2c3c4)` to their three successors on the other levels using the additional links. Then the requestor and the requested nodes use theirs finger tables to locate the resource in theirs respective active rings using Algorithm 1. Algorithm 3 presents the pseudo code for the lookup process executing by node N_j and using additional links (*at nodes on level j*).

Algorithm 3. Lookup acceleration in HPM using additional links.

```
Lookup-AL(Key $c_1c_2c_3c_4$)
1: Begin
2: Send the request Lookup-AL(Key $c_1c_2c_3c_4$) to the three
   successors in rings levels $i$, where $1 \le i \le 4$ and $i \ne j$ using the
   additional links, and execute Lookup(Key $c_1c_2c_3c_4$) on the
   local ring.
3: At the reception of request Lookup-AL(Key $c_1c_2c_3c_4$),
   execute Lookup(Key $c_1c_2c_3c_4$) on this active ring.
4: End.
```

Theorem 2. *The number of hops between any two arbitrary nodes on HPM using additionally links is $1+\log_2(n_i)$ hops, where n_i is the number of nodes in one ring.*

Proof. Send the request `Lookup-AL` (`Key` $c_1c_2c_3c_4$) simultaneously to one node on each level using additional links is achieved in one hop. Execute the request `Lookup` (`Key` ($c_1c_2c_3c_4$) on each ring that receives request `Lookup-AL` is achieved in $\log_2(n_i)$ hops, consequently, the number of hops using additional links is $1+\log_2(n_i)$. □

Acceleration using broadcast mechanism: another technique to accelerate lookup process is to use a broadcast mechanism on each ring, based on the ith part of resource key at each level. We assume that on each ring, nodes are completely connected (*full mesh topology, without finger table*). Thus, the number of control messages generated is $\sum_{i=1}^{4}(n_i-1)$, where n_i is the number of nodes on each ring covered by the request. Even with a broadcast mechanism, the overhead does not have a significant impact on the global performance of the system, as each ring is limited by a maximum of 256 nodes. Algorithm 4 describes the lookup acceleration process based on the broadcast mechanism. In this case, the cost of lookup is four hops.

Algorithm 4. Lookup acceleration in HPM using a broadcast mechanism.

```
Lookup-BM (Key $c_1c_2c_3c_4$)
1: Begin
2:   locate the node (in the same ring) of IP address ($p_1p_2p_3p_4$)
     where $P_i$ is the smallest value greater or equal to $c_i$, by a simple
     broadcast message on the active ring.
3:   IF $\exists c_j$ where $c_j > p_j$ and $j < i$ Then
4:     go to level $(i-1)$ and call lookup-MB (Key $c_1c_2c_3c_4$) (if
       $i > 1$, otherwise, data does not exist)
5:   Else
6:     If the data is present Then
7:       loading the data
8:     Else
9:       go to level $(i+1)$ and call lookup-BM (key $c_1c_2c_3c_4$)
         (if $i < 4$, otherwise, data does not exist)
10: End
```

Theorem 3. *The number of hops between two arbitrary nodes in HPM using broadcast mechanism on each ring is 4.*

Proof. On each ring, the nodes are completely connected, with one hop from any node to any other node. To cover the four levels of HPM, four hops are needed. □

3.5. HPM architecture: extension and generalization

For HPM architecture depending on the number of levels, we consider two cases: four levels in case of IPv4 addressing format and 16 levels in case of IPv6 addressing format, and then we give a generalization for any k levels, both in case of IPv4 and IPv6 addressing format.

- The HPM architecture based on a four levels topology as previously described uses 8 bits for the identifiers space from IPv4 address for each ring. The number of nodes on each ring is 256. The key structure is then $c_1c_2c_3c_4$.
- We use 8 bits from the IPv6 address for each ring, then the maximum number of nodes for each ring is 256, but with a maximum of 16 levels. The key structure becomes $c_1c_2 \ldots c_{16}$.
- The general case corresponds to the usage of k bits from an IPv4 (*resp. IPv6*) address for each ring, then, the maximum number of levels is $32/k$ (*resp. 128/k*). The key structure becomes $c_1c_2 \ldots c_{32/k}$ (*resp.* $c_1c_2 \ldots c_{128/k}$). When $k=32$ (*resp. 128*) using IPv4 (*resp. IPv6*) address format, HPM topology is limited to one ring, as Chord.

HPM topology as opposed to the common architecture for P2P is based on a set of hierarchical rings. When the number of nodes joining and leaving the HPM system increases, the number of rings increases also, but with a controlled size for the finger table (*scalability*). Figure 9 gives an illustration of the HPM architecture in a large network.

3.6. HPM properties

The main properties that characterize HPM are:

- *Scalability*: logarithmic increase of cost of lookup ($O\sum_{i=1}^{4}\log_2(n_i)$, *$n_i$ represents the number of nodes on one ring, it is equal to 256 in case of four levels as described above*) as a function of the number of nodes, even in large scale situation. The finger table size is $2 \times \log_2(n_i)$. This size is compatible with terminals characterized by limited capabilities such as PDA or mobile terminals. As a result, HPM can be supported by terminals with low capabilities, without significant performance degradation.
- *Decentralization*: HPM is completely distributed, thus improving the robustness of the global architecture (*each node is completely equivalent in terms of functionality*), and without any super peers such as in classical hierarchical P2P systems.
- *Load balancing*: distributed hash function spreads keys uniformly and evenly among nodes.

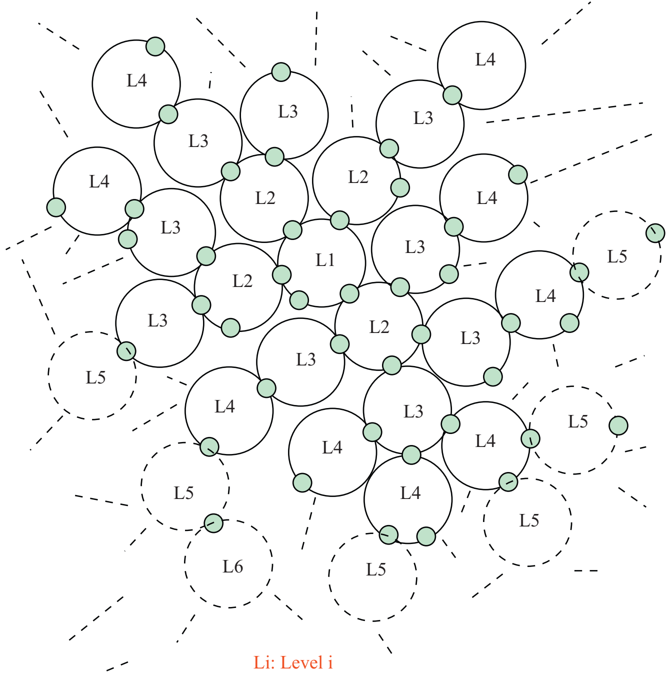


Fig. 9. HPM architecture in a large network.

- *Fault tolerance*: lookup process is active during simultaneous node failures. The additive links guarantee and keep a complete connected system.
- *Cost*: cost of lookup process in HPM is $O(\sum \log_2(n_i))$, and provide better performance than Chord protocol as: $O(\sum \log_2(n_i)) < O(\log_2(n))$, where n is the number of nodes in Chord system, and n_i the number of nodes on one ring in HPM architecture. Some other mechanisms for lookup acceleration (*broadcast and additional links as described above*) are possible without significant performance degradation or extra overhead (see Section 4).

Although NATs and firewalls traversal are not considered in this paper, they constitute a common problem for HPM but also for all kind of P2P applications. For HPM, a variety of existing techniques such as Universal Plug and Play (UPnP), Simple traversal UDP through Network Address Translators (STUN), Application Level Gateway (ALG) or UDP/TCP hole punching can be used and implemented (Hu). However, this is not the subject of this paper; it will be investigated in future work.

4. HPM performance evaluation

In this section, we present some performance evaluations through analytical[1] and simulation. HPM architecture implementation[2] has been carried out using a java platform. The metrics that are defined to evaluate HPM performances are:

- *Cost of lookup*: it is defined as the number of hops or delay needed for resource localization.
- *Size of data structure*: the impact of data structure stored in each node.
- *Number of rings at each level*: the physical proximity on rings at the lower levels is emphasized.
- *Number of rings and levels for HPM with IPv6 and IPv4 address format*: adequate number of level configuration in HPM for IPv6 and IPv4 address format.

We present the HPM architecture characteristics (*the number of levels and the number of rings at each level when using IPv4 or IPv6 address format*). We compare HPM with some representative DHTs in terms of cost of lookup and routing information stored on each node for routing and architecture management. We analyze the lookup in terms of number of hops; but also in terms of lookup delay.

4.1. HPM architecture characteristics

In HPM architecture, each node has a few neighboring nodes ($2 \times O(\log_2(n_i))$, $n_i \le 256$), and uses a greedy routing decisions. The cost of lookup can be optimized up to four hops, using a broadcast mechanism.

For HPM, Figs. 10 and 11 show the number of rings at each level, both HPM with 4 and 16 levels are considered. Level 4 (*the last level in case of four levels*) contains a maximum number of ring (256^3), with a maximum of 256 nodes on each one. At this level, nodes are closed to in terms of physical and logical proximities. Thus, and this is an important characteristic for HPM, the lookup delay is controlled. As an example, nodes with IP addresses: 176.16.10.11, 176.16.10.12, 176.16.10.23, 176.16.10.43, 176.16.10.65, … are on the same ring at level 4. Because HPM architecture takes into consideration the physical proximity, it can be considered as a good candidate for supporting real time applications.

Figures 12 and 13 show the number of levels depending on the number of bits used for nodes identifiers space, considering both IPv4 and IPv6 address format. As an example, using 8 bits in IPv4 address formats (*resp. IPv6 address format*), the number of levels in HPM is 4 (*resp. 7 levels*). Thus, the number of levels has an important impact on cost of lookup performance and finger table size, but also on architecture complexity and topology stabilization. When a 128 bits identifier space is used (*IPv6 address format*) or 32 bits (*IPv4 address format*), HPM architecture is limited to one ring, and then is equivalent to Chord architecture. The number of levels can be controlled through the size (*number of bits*) of the node identifier. The number of levels decreases as the number of bits for node identifier increases and vice versa.

4.2. HPM vs other DHTs

This sub-section resumes HPM performance through a comparison with some representative DHTs. Most of the existing protocols are considered scalable and fault tolerant, but with a cost of lookup that is optimized for HPM, as compared to Chord for example ($\sum \log_2(n_i) \le \log_2(n)$). As shown in Table 2, when HPM is combined with a broadcast mechanism on each ring, the cost of lookup is significantly improved (*four hops*). Each ring in HPM architecture represents the Chord architecture; rings are linked with relay node. HPM architecture is more complex in terms of cost of stabilization than Chord architecture, particularly in case of unstable nodes (e.g. *short lifetime periods*). However, the cost of lookup is more optimized in HPM than in Chord especially for large scale network. Consequently, HPM is more efficient and performance for large scale and real time applications.

4.3. Finger table size evaluation

Figure 14 shows the finger table size for both Chord (*which is considered as a benchmark for structured P2P architecture*) and HPM. When the number of nodes reaches 2^{16}, the finger table size in HPM becomes steady, and equal to $2 \times \log_2(n_i)$ with $n_i \le 256$. With Chord, the finger table increases logarithmically. The finger

[1] Using Matlab V7.
[2] 1.5 GHz of CPU and 256 Mb of RAM.

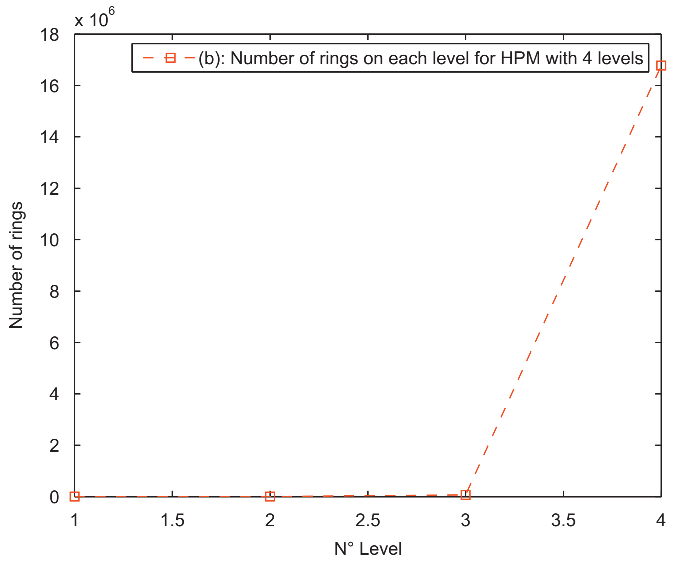


Fig. 10. Number of rings on each level for HPM with four levels.

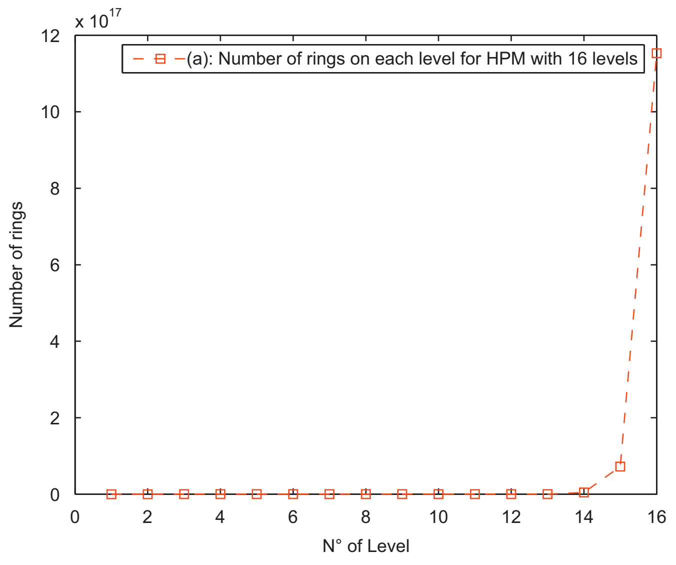


Fig. 11. Number of rings on each level for HPM with 16 levels.

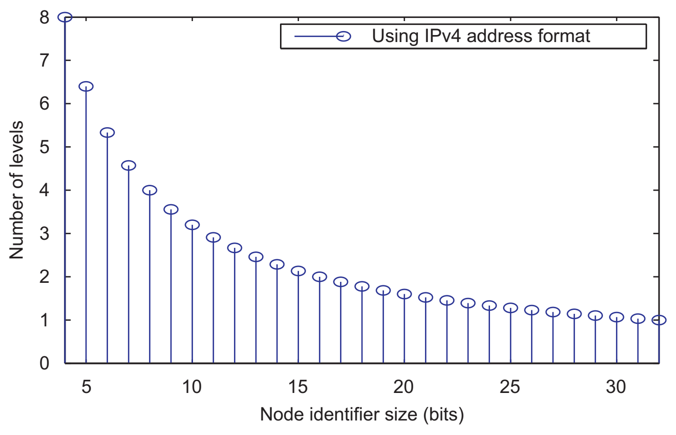


Fig. 12. Number of levels in HPM as a function of the number of bits used for nodes identifiers space (*case IPv4*).

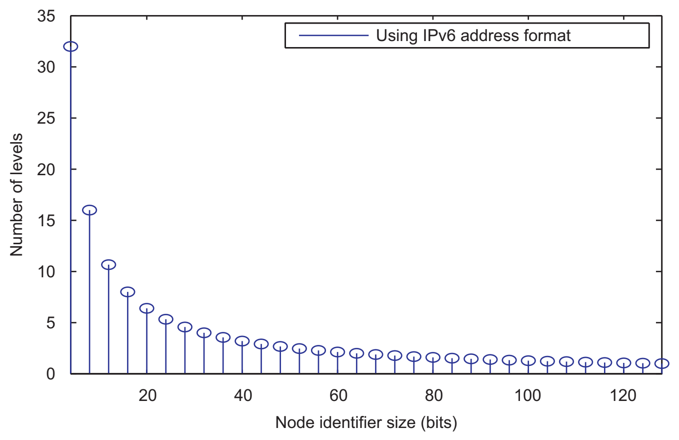


Fig. 13. Number of levels in HPM as a function of the number of bits used for node identifier spaces (*case IPv6*).

table size is a key factor for scalability, especially in critical environments (e.g. *mobile or sensor networks*). Nodes with limited capabilities (e.g. *PDA*) can participate efficiently in HPM, while for Chord the impact on performance can be significant, when both routing process and topology maintenance are supported simultaneously by limited capabilities nodes.

4.4. *Cost of lookup in HPM*

Figure 15 shows that the cost of lookup is significantly reduced compared to Chord. For 256 nodes, the lookup in both HPM and Chord is eight hops, and for 1024 nodes, the cost of lookup is 10 hops for Chord, and it is equal to nine hops in HPM. For 32,768 nodes, the cost of lookup is 15 hops for Chord while it is 13 for HPM. In basic HPM, the cost of lookup is $O(\sum \log_2(n_i))$ and in Chord, it is equal to $O(\log_2(n))$, where n is the number of nodes in Chord, and n_i is the number of nodes on one ring for HPM. This shows that HPM is more efficient than Chord in terms of number of hops for lookup.

Figure 16 shows the cost of lookup for HPM, when 16 levels and four levels are considered, without any acceleration mechanisms (*broadcast or additional links*). As shown, for 69,904 nodes, the cost of lookup on HPM with four levels is 15 hops, and it is equal to 14 hops when 16 levels are considered. As a result, when the number of levels increases; both the cost of lookup and the finger table size decrease.

Figure 17 measures the cost of lookup in terms of latency for both HPM and Chord. As an example, for 60 nodes, the lookup delay is 10 ms for HPM (*resp. 10 ms for Chord*), and for 100 nodes, the delay for a resource lookup from any requestor node to the requested one is 13 ms on average (*resp.*15 ms *for Chord*). The lookup increases in a logarithmic way, thus improving the scalability while providing a rapid convergence for HPM. HPM architecture provides an optimized lookup mechanism combined with a rapid convergence. This is essentially due to one of the main HPM characteristics that bring together nodes that are close to in terms of physical and logical proximities.

5. Conclusion and future work

Peer-to-Peer networks allow resources discovery and localization from a resource identifier (e.g. *key words*). In the context of today networks (e.g. *social networks*) where resources but also services tend to be created and provided by the end user, P2P can be used for improving communication process, optimizing

Table 2
A comparison of some representative P2P DHTs.

Scheme	Base network	Routing table size	Cost of lookup
Chord (Stoica et al., 2003)	Ring	$O(\log(n))$	$O(\log(n))$
CAN (Ratnasamy et al., 2001)	d-Dimensional Cartesian space	$O(d \times n^{1/d})$	$O(d)$
Pastry (Rowstron and Druschel, 2001)/**Tapestry** (Zhao et al., 2004b)	Hypercube	$O(\log(n)$	$O(\vert L\vert)+O(\vert M\vert)+O(\log(n))$
Viceroy (Malkhi et al., 2002)	Butterfly	$O(\log(n))$	7
ABC (Xu, 2005)	Clustering	$O(\log(n))$	$O\left(\frac{\log(n)}{\log(\log(n))}\right)$
HPM	Hierarchical rings	$2 \times O(\log(n_i)$	$O(\sum \log(n_i))$
HPM with additional links	Hierarchical rings	$2 \times O(\log(n_i)+3$	$O(1+\log(n_i))$
HPM with broadcast	Hierarchical rings	$2 \times n_i$	4

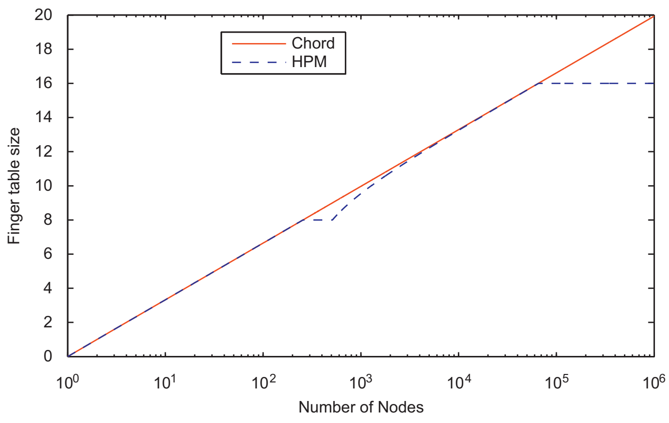


Fig. 14. Finger table size for both Chord and HPM.

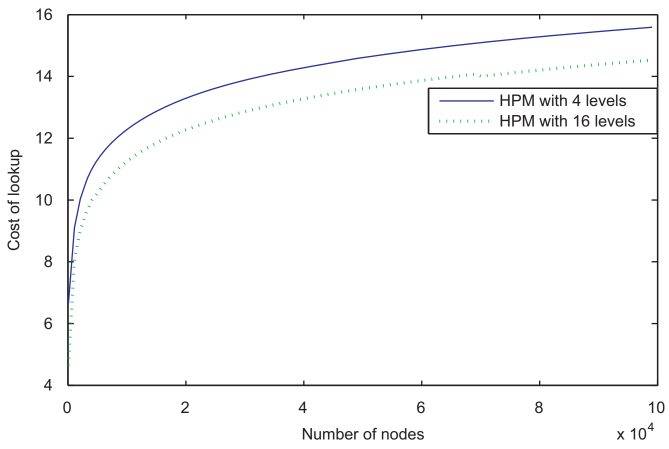


Fig. 16. Cost of lookup (*number of hops*) for HPM with 16 and 4 levels.

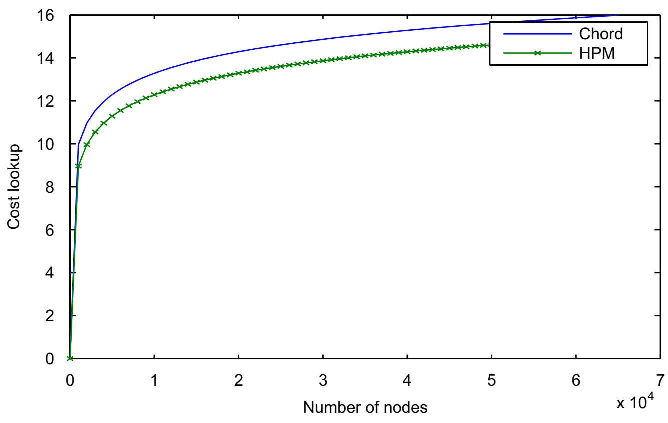


Fig. 15. Cost of lookup (*number of hops*) for Chord and HPM.

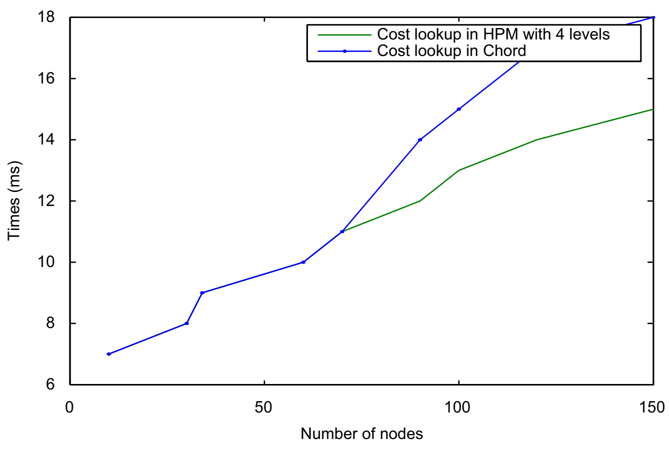


Fig. 17. Cost of lookup (delay): HPM with four levels vs Chord.

resources utilization, and for facilitating distributed information exchange (Schoder and Fischbach, 2005).

Related works show that unstructured P2P networks performance can be improved, by flooding techniques and random walkers. Nevertheless flooding-based search mechanism is not adapted to large scale systems. Although random walkers can reduce flooding by some extent, it also creates significant network overhead, due to the involvement of many forwarding peers. Furthermore, with network flooding and random walkers false negative exist.

Structured P2P networks are organized and based on a controlled topologies. The data placement and lookup algorithms are rigorously defined based on a distributed hash table (DHT). There is no false negative, even in high dynamic context. Due to their potential efficiency, robustness, scalability and deterministic data location, structured networks have been studied intensively these recent years. Nevertheless, optimizing both lookup and overhead costs in such networks, characterized by high scalability and dynamicity that still constitute key issues.

In this context, we propose a new approach that improves resource discovery and location. Our proposed HPM architecture provides this discovery/localization service, based on a complete decentralized architecture, by determining with efficiency the node responsible for storing the requested key's value. The node identifier is simply derived, as it is built from one part of its IP address (*and port number in private networks*), while the resources identifiers are generated by a hashing function from the resource name (*or meta-information*) as key. One of the main characteristics of HPM is the routing optimization at IP level, as it takes into consideration the physical proximity while minimizing the number of hops for lookup process (*cost of lookup*).

In a N-node HPM network, each node maintains routing information for only $2 \times O(\log_2(n_i))$, where $n_i < N$ is the number of nodes on one ring, with a maximum of 256. Also HPM takes into consideration the physical topology and proximity. On rings corresponding to "level 4", nodes are physically and logically closed to, thus reducing significantly the lookup delay. So, the physical proximity increases as the ring level increases. In this way, the cost of lookup in HPM architecture is $\sum_{i=1}^{4}(\log_2(n_i))$. The use of additional links or a broadcast mechanism on the topology can significantly and efficiently accelerate the lookup process, without a significant performance degradation. Performance evaluation and analyses show that results are globally satisfactory; when considering HPM, the cost of lookup is significantly reduced (e.g. *four hops for the cost of lookup when using broadcast mechanism*). HPM uses a simple approach for node identifiers (*compared to SHA-1 or SHA-2*), that are derived from their IP addresses. This provides a certain flexibility and robustness, particularly in dynamique environment. HPM can be also implemented using either IPv4 or IPv6.

P2P networks tend to become a key element for Internet communications, such as legacy applications (e.g. *file sharing*), but also for VoIP (Singh and Schulzrinne, 2004). However, efficient security and trust management constitute a serious concern for P2P. In terms of perspectives, we envision two directions: first, for taking into consideration security aspects, security protocols such as Renuka and Shet (2009) can be combined and extended in HPM context for large peers groups communications. Second, a recent technique for application layer multicast (Amad et al., 2011) can be implemented using the proposed HPM for real time oriented applications.

Acknowledgments

The authors would like to thank Dr B. Rabta from Neuchatel University (*Switzerland*) and Mr M. Saadi from Bejaia University (*Algeria*) for their comments.

References

Aberer K, Alima LO, Ghodsi A, Girdzijauskas S, Haridi S, Hauswirth M. The essence of P2P: a reference architecture for overlay networks. In: Proceedings the 5th IEEE international conference on peer-to-peer computing; 2005. p. 11–20.

Bai X, Liu S, Zhang P, Kantola R. ICN: Interest-based clustering network. In: Proceedings of the 4th international conference on peer-to-peer computing (P2P'04), August; 2004. p. 219–26.

Balakrishnan H, Kaashoek F, Karger D, Moris R, Stoica I. Looking up data in P2P systems. Communication of the ACM 2003;46(2):43–8.

Chao S, Hui L, Feng L, Yan J. Node clustering in the P2P environment. In: Proceedings of the international conference on networking, international conference on systems and international conference on mobile communications and learning technologies (ICNICONSMCL'06), April; 2006. p. 61.

Cramer C, Kutzner K, Fuhrmann T. Bootstrapping locality-aware P2P networks. In: Proceedings of the 12th IEEE international conference on networks; 2004. p. 357–61.

Garces-Erice L, Biersack EW, Ross KW, Felber PA, Urvoy-Keller G. Hierarchical P2P systems. In: Proceedings of ACM/IFIP international conference on parallel and distributed computing (Euro-Par). 2003. p. 1230–39.

Gnutella ⟨http://www.gnutella.com⟩.

Hsiao R, Wang S. Jelly: a dynamic hierarchical P2P overlay network with load balance and locality. In: Proceedings of the 24th international conference on distributed computing systems workshops (ICDCSW'04); 2004. p. 534–40.

Hu Z. NAT traversal techniques and peer-to-peer applications ⟨http://www.tml.tkk.fi/Publications/C/18/hu.pdf⟩.

Joung Y-J, Lin Z-W. On the self-organization of a hybrid peer-to-peer system. Journal of Network and Computer Applications 2010;33:183–202.

Joung Y, Wang J. *Chord*^{2}: a two-layer Chord for reducing maintenance overhead via heterogeneity. Journal of Computer Networks 2007;51(3):712–31 (Elsevier).

Lee J, Lee H, Kang S, Kim SM, Song J. CISS: an efficient object clustering framework for DHT-based peer-to-peer applications. Journal of Computer Networks 2007;51(4):1072–94 (Elsevier).

Li J, Vuong S. An Efficient clustered architecture for P2P networks. In: Proceedings of the 18th international conference on advanced information networking and application (AINA'04); 2004. p. 278.

Lua EK, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Survey and Tutorial 2004;7(March (2)):72–93.

Malkhi D, Naor M, Ratajczak D. Viceroy: a scalable and dynamic emulation of the butterfly. In: Proceedings of the 21st annual symposium on principles of distributed computing (PODC); 2002. p. 183–92.

Martinez-Yelmo I, Cuevas R, Guerrero C, Mauthe A. Routing performance in a hierarchical DHT-based overlay network. In: Proceedings of the 16th Euro-micro conference on parallel, distributed and network-based processing; 2008. p. 508–15.

Maymounkov P, Maziére D. Kademlia : a peer to peer information system based on the XOR metric. In: Proceedings of the 1st international workshop on peer-to-peer systems (IPTPS) ⟨http://kademlia.scs.cs.nyu.edu⟩; 2002.

Miasnikov AD, Rome JE, Haralick RM. A hierarchical projection pursuit clustering algorithm. In: Proceedings of the 17th international conference on pattern recognition (ICPR'04), August; 2004. p. 268–71.

Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, et al. Peer to peer computing survey. Technical report. HP Laboratories Palo Alto, HPL-2002-57, March; 2002.

Mourad Amad, Ahmed Meddahi, Djamil Aïssani. Gilles Vanwormhoudt. GPM: a generic and scalable P2P model that optimizes tree depth for multicast communications. International Journal of Communication Systems 2012;25(4): 491–514.

Napster ⟨www.napster.com⟩.

Peer-to-Peer Working Group. Bidirectional peer-to-peer communication with interposing Firewalls and NATs. White Paper; 2001.

Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content addressable network. In: ACM SIGCOMM; 2001. p. 161–72.

Renuka A, Shet KC. Cluster based group key management in mobile ad hoc networks. International Journal of Computer Science and Network Security 2009;9(April (4)):42–9.

Rowstron A, Druschel P. Pastry: a scalable, decentralized object location and routing for large scale peer to peer systems. In: Proceedings of the 18th IFIP/ACM international conference on distributed systems plateforms (Middleware 2001). Heidelberg, Germany, November; 2001. p. 329–50.

Schoder D, Fischbach K. The peer-to-peer paradigm: minitrack introduction. In: Proceedings of the 38th Hawaii international conference on system sciences, IEEE internet computing; 2005.

Shena H, Xua CZ, Chenb G. Cycloid: a constant-degree and lookup-efficient P2P overlay network. Journal of Performance Evaluation 2006;63(3):195–216 (Elsevier).

Shin K, Lee S, Lim G, Yoon H, Ma JS. Grapes: topology-based hierarchical virtual network for peer-to-peer lookup services. In: Proceedings of the international conference on parallel processing workshops (ICPPW'02); 2002. p. 159–66.

Singh K, Schulzrinne H. P2P Internet telephony using SIP. Technical report. Department of Computer Science, Columbia University; 2004.

Steinmetz R, Wehrle K. Peer to peer systems and applications. Lecture Notes in Computer Science, vol. 3485. Springer; 2006.

Stoica I, Morris R, Liben-Nowell D, Karger D, Kaashoek MF, Dabek F, et al. Chord: a scalable peer-to-peer lookup service for Internet application. IEEE/ACM Transactions on Networking 2003;11(January (1)).

Tong X, Zhang D, Yang Z. Efficient content location based on Interest-cluster in peer-to-peer system. In: Proceedings of the 2005 IEEE international conference on e-business engineering (ICEBE'05), October; 2005. p. 324–31.

Xu X. ABC: a cluster-based protocol for resource location in peer-to-peer systems. Journal of Parallel Distributed Computing 2005 (Elsevier).

Zhao BY, Kubiatowicz J, Joseph A. Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 2004a;22(January (1)):41–53.

Zhao BY, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD. Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 2004b;22:41–53.

Zoels S, Despotovic Z, Kellerer W. Cost-based analysis of hierarchical DHT design. In: Proceedings of the 6th IEEE international conference on peer-to-peer computing; 2006. p. 233–9.

Zoels S, Despotovic Z, Kellerer W. On hierarchical DHT systems—an analytical approach for optimal designs. Journal of Computer Communication 2008;31(3):576–90 (Elsevier).