Wireless Pers Commun
DOI 10.1007/s11277-015-2851-y

De Bruijn Graph Based Solution for Lookup Acceleration and Optimization in P2P Networks

Mourad Amad[1] · Djamil Aïssani[1] · Ahmed Meddahi[2] · Makhlouf Benkerrou[3] · Farouk Amghar[3]

© Springer Science+Business Media New York 2015

Abstract There is no doubt that P2P traffic mainly video traffic (e.g. *P2P streaming, P2P file sharing, P2P IPTV*) increases and will represent a significant percent of the total IP video traffic (*80 percent by 2018 of the global IP traffic according forecasts*). Peer-to-peer (P2P) is based on some main concepts such as mutualization of resources (e.g. *data, programs, service*) at Internet scale. It is also considered as one of the most important models able to replace the client-server model (e.g. *for media streaming*). Nevertheless, one of the fundamental problems of P2P networks is to locate node emplacements or resources and service location. Localisation problem is critical as there is no central server and churn rate can be high in some environments (*high dynamicity*). Lookup optimization in terms of number of hops or delay is not well considered in existing models, and still represents a real challenge. In this context and according to their specific characteristics and properties, De Bruijn graph based solutions constitute good candidates for lookup optimization. In this paper, we propose a new optimized model for lookup acceleration on P2P networks based on De Bruijn graph. Performance evaluations and simulation results show that our proposed approach is performant, compared to the main existing model.

✉ Mourad Amad
amad.mourad@gmail.com

Djamil Aïssani
lamos.bejaia@univ-bejaia.dz

Ahmed Meddahi
ahmed.meddahi@telecom-lille.fr

Makhlouf Benkerrou
mbenkerrou2001@yahoo.fr

Farouk Amghar
farouk.amghar@univ-bejaia.dz

[1] LaMOS Research Unit, Faculty of Exact Sciences, University of Bejaia, 06000 Bejaïa, Algeria

[2] Institut Mines Telecom, Telecom Lille, Lille, France

[3] Bejaia University, Bejaïa, Algeria

Published online: 30 June 2015

Keywords P2P · Resource localization · Lookup acceleration · De Bruijn graphs

1 Introduction

P2P lookup optimization and acceleration consists of relaying rapidly each node with others, with few intermediaries, while maintaining only a minimum of contacts for each node. Graph theory allows to formulate and model this problem. On one hand, a network can be represented as a graph with: nodes in the graph correspond to nodes in the network and the arcs correspond to contacts of the node. On the other hand, lookup services is a fundamental process in each P2P network. It consists of finding resources stored in any node of the network or the node profile (*user*) from any other node. This is the most and the main executed function. However, lookup optimization (e.g. *number of hops, delay*) is a critical issue. In this representation, the optimized path corresponds to the connected graph with optimized degree and diameter. In order to solve this problem, many types of solutions have been proposed in the literature, each one uses a particular graph (e.g. *ring, hierarchical ring, d-tore,…*). However, lookup acceleration and optimization in terms of number of hops (*virtual parameter*) and delay (*physical parameter*) is a real challenge, particularly for De Bruijn graph based solutions where the graph presents some relevant characteristics such as degree and diameter.

Many classifications of P2P networks have been proposed in the literature. The first one considers the degree of centralization. In this classification, three architectures have been considered: centralized such as Napster,[1] decentralized such as Gnutella[2] and hybrid such as KaZaa.[3] The second classification considers the structuration degree of the network topology. In this classification, we distinguish two classes: structured architecture like CAN [1] and unstructured architecture like in [2].

The number of hops for lookup and localization service on application layer in any peer to peer network is higher than the corresponding number of hops in physical layer. So, routing acceleration and optimization is still a challenge, particularly for real time applications such as P2P-IPTV and VoIP.

Resources/nodes localization in peer to peer networks works independently of the underlying layer and constitutes a logical network, requiring a mechanism for naming both for data and peer. Distributed hash tables (DHTs) are examples of routing layers that implement such an approach (e.g. *Chord* [3], *Pastry* [4], *HPM* [5] *and* [6]). However, from maintenance stabilization point of view, the network degree needs to be optimized. Given this, recent mechanisms based on specific graphs (e.g. *Skip graph* [7], *Knodel graph* [8], *Kayley graph* [9], *Pancake graph* [10], *Skip List* [11]…) have been investigated, particularly, De Bruijn graph based solutions [12] which has led to the development of systems as Koorde [13] and D2B [12]. De Bruijn graph constitutes a good candidate for lookup acceleration and optimization in large scale P2P networks, because it provides a constant degree with a logarithmic diameter.

In this paper, we propose a scalable solution for lookup acceleration and optimization for structured peer to peer networks based on De Bruijn graph. In the proposed solution, in

[1] www.napster.com.

[2] www.gnutella.com.

[3] www.kazaa.com.

order to accelerate and simplify the lookup process, some specific tasks are executed locally.

The rest of this paper is organized as follow: In Sect. 2, we give a background regarding De Bruijn graph and related works on lookup service based on De Bruijn graph in the context of P2P networks. In Sect. 3, we present our solution for lookup acceleration and optimization. Performance evaluation of the proposed model is given in Sect. 4. Finally, we conclude and give perspectives, particularly for the application of the proposed model under high churn conditions.

2 Background and Related Works

In this section, we present the fundamental concepts of De Bruijn graph with a focus on concepts that are used in our proposed solution. Furthermore, a critical analysis of some principal existing solutions based on De Bruijn graph is also given in order to position, evaluate and compare our proposed solution.

2.1 De Bruijn Graph Concepts

De Bruijn graph denoted $B(d, D)$ is a directed graph whose nodes are words of length D on an alphabet of size d. There is an arc from node x to node y if and only if the $D-1$ first letters of y correspond to $D-1$ last letters of x. As a fairly obvious way, the parameter d is the degree of the graph [14]. Figure 1 shows an example of De Bruijn graph B (2, 3). An arc from node (bba) to node (bab) exists, because both nodes have a common prefix and suffix (ba).

2.1.1 Eulerian Circuit

It is an Euclidean path that passes once and only once by each arc of the considered graph. An Eulerian circuit is a path whose extremities are confused.

2.1.2 Words in De Bruijn Graph

In some cycles of bits, by setting a size n, we found all sets of n bits exactly once in this circuit. A cycle or a sequence of De Bruijn of order n with k letters is a word of length k^n containing circularly all words of length n over the alphabet $\{0, 1, \ldots k-1\}$. Thus, this cycle provides a way to enumerate all words of length n. As an example, for $k = 2$ and

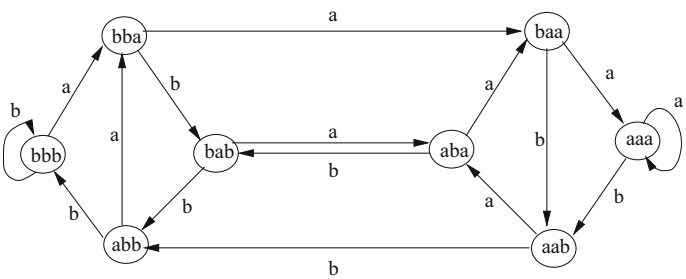

Fig. 1 De Bruijn graph B (2, 3) of order D = 3, $A = \{a, b\}$

$n = 3$, the word $x = aaababbb$ is a De Bruijn cycle of order 3. Indeed, the eight binary words of length 3 appear successively in x organized circularly in sequence (*aaa, aab, aba, bab, abb, bbb, bba, baa*). The first two concepts (*euclidean circuit and word of De Bruijn*) consider that finding a De Bruijn sequence is equivalent to finding an Eulerian circuit.

For each node x, both degrees $d^-(x)$ and $d^+(x)$, corresponding to the numbers of incoming and outgoing arcs are equals and therefore an Eulerian circuit exist [14].

The De Bruijn graph of order k for an alphabet $A = \{a1, \ldots ak\}$ of size n is defined as follows:

1. Nodes are the set of all words of length n on A,
2. Each node has k outgoing edges (*arcs*) labeled by the letters of A,
3. Successors of nodes are obtained by removing its first letter and adding one of k possible letters at the end.

Figure 1 represents a De Bruijn graph of order 3.

Algorithm 1 allows to browse the graph illustrated in Fig. 1 without passing twice through the same arc. As an example, from node bbb, the Eulerian Circuit is: {**bbb**-(a)-bba-(a)-baa-(a)-aaa-(a)-aaa-(b)-aab-(a)-aba-(a)-baa-(b)-aab-(b)-abb-(a)-bba-(b)-bab-(a)-aba-(b)-bab-(b)-abb-(b)-bbb-(b)-**bbb**}.

Algorithm 1 De Bruijn Graph run through Algorithm

1. **Begin**
2. Choose a starting node,
3. From the current node, first out by the arc labelled a, and by the arc labelled b only if it is already out in a,
3. Stop when continuing is not possible (*it is already out by a and b*),
4. **End.**

2.2 Related Works on De Bruijn Graph Based Solutions

De Bruijn graphs have been proposed for many applications [15] but also for network routing. P2P community is interested to exploit these graphs and their applications as part of DHTs. Four principle infrastructures have emerged: D2B [12], Koorde [13], Distance Halving [16] and DH-DHT [17].

2.2.1 D2B

The D2B [12] with two-dimensional employs the set $K = \{0, 2^m - 1\}$ as a principal area, also seen as a set of binary string s of length m. All participants in D2B employ the same function H that hashes the resource IDs in K. D2B nodes have also an identifier represented as a binary string, but the length of ID is less than or equal to m. Thus, there are at most 2^m nodes in D2B.

The value of a node u with identifier $x_1 \ldots x_k$ is defined by the function $val(u) = 2^{m-k} * \sum_{i=1}^{k} (x_i * 2^{k-i})$, where each node is responsible for the keys in the range $[val(u), 2^{m-k} * \sum_{i=1}^{k} (x_i * 2^{k-i} + 1) - 1]$.

Binary representation of a key $k_1 \ldots k_m$ is referenced by the node of identifier $x_1 \ldots x_k$ if and only if $x_1 \ldots x_k$ is a prefix of $k_1 \ldots k_m$. From the routing table viewpoint, each peer D2B

Table 1 Routing table entries for a peer u with identifier $x_1 \ldots x_k$ in D2B

Links	Identifiers
Children	$X_2 \ldots X_j$ with $j \leq k$ or $X_2 \ldots X_k Y_1 \ldots X_l$, with $1 \leq l \leq m - k + 1$
Parents	$\alpha X_1 \ldots X_j$ with $\alpha \in 0, 1$ and $j \leq k$ or $\beta X_1 \ldots X_k Y_1 \ldots Y_l$ with $\beta \in 0, 1$ and $1 \leq l \leq m - k - 1$
Brothers v and w	w big brother of v if it has the smallest value $val(w)$ greater than v
Children of u	w little brother of v if it has the largest value $val(w)$ lower than v

maintains three types of links (*entries*) to successors (*children, parents, and brothers*). Table 1 shows the formal description of the identifiers for these three types of entries [12].

2.2.2 *Koorde*

For simplification purpose, let us consider the simple scenario (*but theoretical*) where all identifiers designate a node. Algorithm 2 illustrates the routing process in Koorde; where n denotes the current node in the lookup process, n' the successor, k the desired identifier, *kshift* corresponds to k but shifted by the number of previous iterations. o is a shift operator in the considered base (e.g. $d = 2$): $xoy = 2 \times x + y mod 2^D$. The initial call made by the node n that seeks resource k is *n.search* (k, k). Initially, the algorithm places the first letter of k in the last position of the current node, then the first two letters in the last two positions, until giving the complete identifier k. The identifiers are with length D and the complexity is with D steps [13].

Algorithm 2 First Routing Algorithm with Koorde

(n.search(k,kshift))
1. Begin
2. if (k=n) **then**
3. return (n)
4. Else,
5. $n' \leftarrow nofirstChar(kshift)$;
6. return (n'.search(k, kshift o O));
7. Endif.
8. End.

In practice, peer-to-peer systems don't take benefits from the ring topology, in order to limit the collision of peer identifiers. Consequently, the above presented theoretical method can't be considered, because in practice some nodes do not exist or can disappear. In fact, unlike Chord [3], each node has a fixed number of contacts in Koorde. So, without theses contacts, the routing process can not be executed. The real process of routing in Koorde will therefore simulate a perfect routing, while not passing just only by existing nodes. Given this, each node n should get two additional contacts: its successor on the ring denoted *succ* and the predecessor of $2n$ denoted p (*equal to 2n if it exists*) [14]. Algorithm 3 illustrates this process.

Algorithm 3 Real Routing Algorithm with Koorde

```
(n.search(k,kshift,i))
1. Begin
2. if (k ∈]n, succ]) then
3. return n.succ
4. else
5. if (i ∈]n, succ]) then
6. return p.search(k,kshift o 0,i o firstChar(kshift))
7. else
8. return succ.search(k,kshift,i)
9. EndIf
10. EndIf
11. End.
```

Algorithm 3 presents an extension of the previous algorithm, taking into consideration the changes and improvements discussed above. A node n looking for a given resource k calls the lookup function with the appropriate parameters (k, k, i). In this function, i represents the current imaginary node of perfect routing that corresponds to the current node in the Algorithm 2.

The basic idea consists of maintaining as current and real node n, the predecessor of imaginary node i. At each step, if n does not verify this property, we use the successors to reach this property. Then, carried out a routing step as in Algorithm 2, shifting node i with n letters: perfect routing arrives at node $2i$ or $2i + 1$, real routing arrives at the node p (*that precede* $2n$). Since n was close to i, the probability to have p close to $2i$ is high. This process is still running until i equal to the requested resource k. At this point, n is the real node, it is the predecessor of i, and thus its successor is also the successor of k, therefore, it gets the data. From the standpoint of complexity, this algorithm has D real routing steps (*equivalent to the Algorithm 2*); the number of steps using the successor can be estimated to $2D$. So, the complexity is equal to $3D = O(log_2 N)steps$ [14].

2.2.3 DH-DHT

The Distance Halving (DH-DHT) [17] is another system based on the De Bruijn graph such as Koorde. However, graph establishment is different. The DH is based on an approach called the continuous discrete approach to the construction of graphs. To establish a De Bruijn graph with this approach, mark space is normalized in a continuous space represented by the interval [0, 1[. Nodes are points in this interval. Each node y has two edges, a left edge $l(y)$ and a right edge $r(y)$ where, $l(y) = \frac{y}{2}$ and $r(y) = \frac{y}{2} + \frac{1}{2}$ [18].

Given the set of points and their edges, a discretization step is made to establish the graph. The set of points are denoted by $\overrightarrow{x}$. Points of $\overrightarrow{x}$ divide the space into n segments. The segment of a point $x_i, S(x_i) = [x_i, x_{i+1}), (i = 1..n - 1)$ and $S(x_n) = [x_{n-1}, 1)U[0, x_1)$. If a node y has an edge that belongs to a segment of node z, then there is an edge in the discrete graph between y and z. We can also note that the segments are defined in order to obtain a circular space mark.

The motivation of using this type of graph is that for each node, the space is divided into two intervals and pointers to two other nodes are kept. The first one is in the middle of the left interval and the second one is in the middle of the right interval. Figure 2a shows the indicators of all nodes in a DH-network of size N = 8. Figure 2b shows the paths of all possible destinations from node 1.

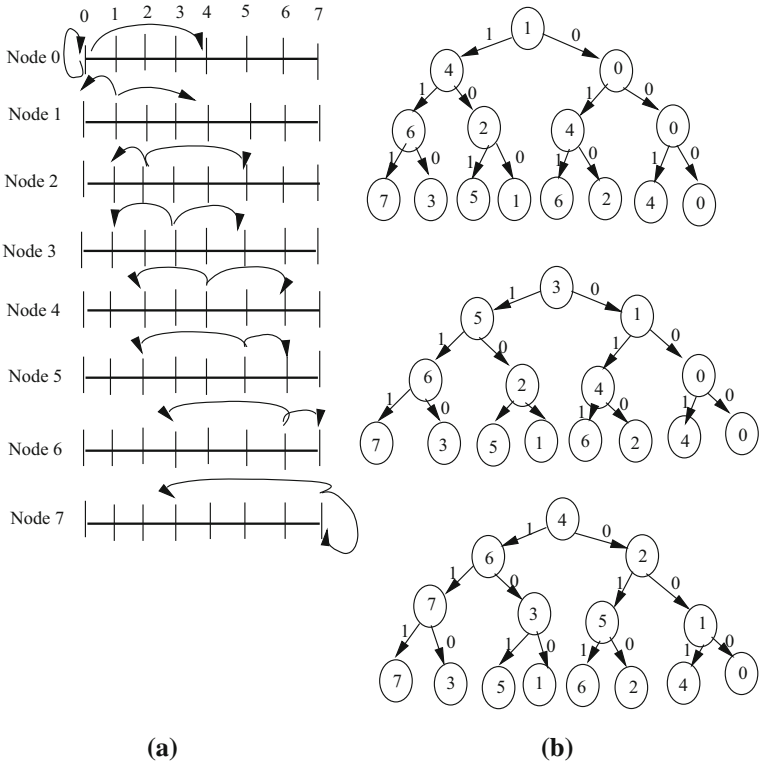

Fig. 2 **a** Indicators of all nodes in a complete Distance-Halving of network where N = 8 and **b** paths from node 1 to other nodes

The lookup request is sent to the specified node near the left edge to match a 0 and the right edge to match a 1. The path length of lookup process is then equal to $O(\lg_2(n))$ [18].

A new node n joins a DH-network by searching a node s such that n belongs to S (s). n then uses s for lookup its left and right edges. from the construction of DH, a node can easily learn which nodes can be reached. Therefore, a node can easily compute nodes to be updated and notify them of the existence of n. Updating the other nodes that leave the DH-network, is made in the same manner.

The DH identifies the failure problems that can lead to a disconnected graph and recommends an additional state of $O(\lg(n))$ indicators. This is compliant with the Koorde principle. The main advantage of Koorde is to have a graph with constant degree. However, this property is compromised if the fault tolerance is considered. With a logarithmic degree, these types of graphs can provide a diameter of $\frac{\lg(n)}{\lg(\lg(n))}$ [18].

Because lookup optimization and acceleration is a critical and it is also the most executed function in P2P networks, it should be limited and controlled. In the following, we propose a novel hybrid solution based on some characteristics of D2B, DH-DHT and Koorde methods in order to accelerate this function.

3 Proposed Solution

In this paper, we propose a new hybrid P2P architecture based on De Bruijn graph. It combines advantages of DH-DHT, D2B and Koorde models in order to optimize and accelerate lookup process while minimizing the architecture management complexity, some parts of the lookup process are locally executed.

3.1 Problem and Motivations

Routing (*lookup*) in P2P networks is done at the application layer and therefore the associate number of hops at the network layer is potentially larger. So, the main objective in the P2P routing process is to reduce as much as possible the number of hops for lookup acceleration and optimization. For Koorde protocol [13], routing is based on the notion of left shift. For example, if the node 0010 requests to reach the node 1011, it will do the following shift operations: $0010 \rightarrow 0101 \rightarrow 1010 \rightarrow 0101 \rightarrow 1011$. So, the cost lookup is four hops. However, we can notice that the nodes 0010 and 1011 have a common string 10, which is the suffix and prefix of nodes 0010 and 1011 respectively, so only two shifts are needed to reach the node 1011 from node 0010.

The proposed hybrid solution consists in combining and preserving the benefits of existing methods previously presented DH-DHT, D2B and koorde. So, we focus on minimizing the number of hops for routing process.

3.2 Functional Principle

To route a lookup request from a node to another in D2B [12], we consider the largest common string between these two nodes, which is the suffix of the first and prefix of the second node. DH-DHT also uses the principle of De Bruijn graph in its topology, so if we exploit the principle illustrated in Fig. 2a, we get the graph represented in Fig. 3, which is a De Bruijn graph B (2, 3) with right shift between nodes. As our solution is based on the De Bruijn graph with right shift, so the routing of a request from a node $X = x1\ldots xD$ to a node $Y = y1\ldots yD$, consists of searching the string s which is a prefix of X and a suffix of Y, instead of searching the longest string s with a suffix of X and a prefix of Y. To implement this solution, we define alpha (α), a new shift operator in the base d (e.g. $d = 2$) as follows:

$$x\alpha y = xdiv2 + y * 2^{D-1}. \tag{1}$$

3.3 Elimination of the Common String Between Source and Destination

Algorithm 4 illustrates the process that removes the common chain (*string*) in the second node (*requested*), which is the prefix of the first node and suffix of the second one, to minimize the number of hops in the routing algorithm (*Algorithm 5*). Algorithm 4 is locally executed by the current requestor node. In fact, the lookup is significantly accelerated.

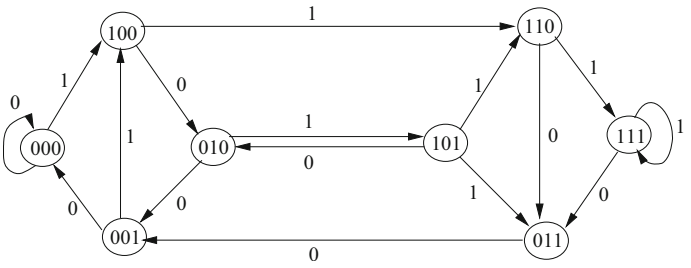

Fig. 3 De Bruijn graph with right shift

Algorithm 4 Common String Elimination Procedure

```
1. Begin
2. n1=n; k1 = k;
3. While (D > 0) do
4. BeginWhile
5.  If (LastChar(k1)! = LastChar(n1)) Then
6.  Begin
7.  n1 = n1 Div 2;
8.  D=D-1;
9.  end
10.  H=D; n2 = n1;
11.  While (LastChar(k1) == LastChar(n2) and (H > 0))
do
12.  begin
13.  n2 = n2 Div 2;
14.  k1 = k1 α 0
15.  H=H-1;
16.  end
17.  If (H > 0) then
18.  k1 = k; n1 = n1 Div 2; D = D - 1;
19.  else
20.  D=0;
21. endWhile
22. if (h==0) then
23. k =k1;
24. End.
```

To understand the benefit of this procedure, we execute it by considering the two following nodes: source node $n = 1000$ and destination node $k = 1110$. The details of each step are given in Table 2.

- Make, $n1 = 1000$ and $k1 = 1110$;
- We have: LastChar (k1) = LastChar (n1) = 0, so the condition (if) is not verified;
- In the second loop at line N9,
- We have n1 = 100, k1 = 0111 and D = 3;
- We have: LastChar (k1) = 1 and LastChar (n1) = 0, then the condition `while` is not verified;
- Outgoing from the `while` loop, we have D = 3, so, k1 is equal to 1110;
- Since $D > 0$, the `while` loop is re-executed, we have n1 = 100, k1 = 1110, thus: LastChar (k1) = LastChar (n1) = 0, then the condition (if) is not verified;

Table 2 Algorithm 4 step by step

n2	n1	n	k1	k	D	H
1000	1000	1000	1110	1110	4	4
100	100		0111		3	3
			1110		3	3
100			0111		0	3
10	10		1110		2	2
10						2
1			0111			1
0			0011	0011	0	0

- Jumped into the second loop,

- We have n1 = 10, k1 = 0111 and D = 2;
- We have: LastChar (k1) = 1 and LastChar (n1) = 0, then the condition is not verified;
- After execution of the loop, we have D = 2, then k1 is equal to 1110;
- Since $D > 0$, we return into the first loop, we have n1 = 10, k1 = 1110, thus: LastChar (k1) = LastChar (n1) = 0, then the condition (if) is not verified;

- Jumped into the second loop:

- We have n1 = 1, k1 = 0111 and D = 1;
- We have: LastChar (k1) = LastChar (n1) = 1, then we go back into the second loop;
- We have n1 = 1, k1 = 0011 and D = 0.
- We have also, D = 0, so we leave the second loop.
- D = 0, then k1 is equal to 0011 and outgoing from the first loop
- D = 0, then k is equal to 0011.

So, after finishing the execution of the procedure illustrated in Algorithm 4, it is possible to eliminate the common string between n and k.

3.4 Routing Algorithm

Algorithm 5 shows the new routing process that supports the changes described above; n denotes the current requestor node, n' the successor, k is the requested identifier; *kshift* is the result of the process illustrated in Algorithm 4. The initial call made by the node n that search the resource K is *n.lookup* (k, k).

Algorithm 5 uses the procedure illustrated in Algorithm 4 to eliminate the common string, then it places the last digit of *kshift* in the first position of the current node, then the last two positions in the first two, until obtaining all parts of the identifier k. The length of the identifier is D, the complexity is $D - L$ steps (L *is the length of the string s, which is the prefix of the current node and suffix of* k).

Algorithm 5 Proposed Routing algorithm

```
Proc n.lookup(k,kshift)
1. Begin
2. n.compar(kshift )
3. if (k==n) then
4. return n
5. else
6. begin
7. n = n α LastChar(kshift);
8. kshift = kshift α 0;
9. go to 3
10. end
11. end.
```

If we execute the routing algorithm on both nodes $n = 1000$ and $k = 1110$, as illustrated on Fig. 4, we use the elimination procedure, then:

- We have *kshift* = 0011,
- We have $k \neq n$, then, we execute else,
- n is equal to 1100,
- *kshift* is equal to 0001,
- go to instruction in line 3.
- We have $k \neq n$, then we execute the else,
- n is equal to 1110,
- *kshift* will be equal to 0000,
- go to instruction in line 3.
- We have k = n, then routing process is finished. The routing path is: 1000 → 1100 → 1110 (*two hops*), instead of path: 1000 → 0100 → 1010 → 1101 → 1110 (*four hops*) in the existing Koorde method.

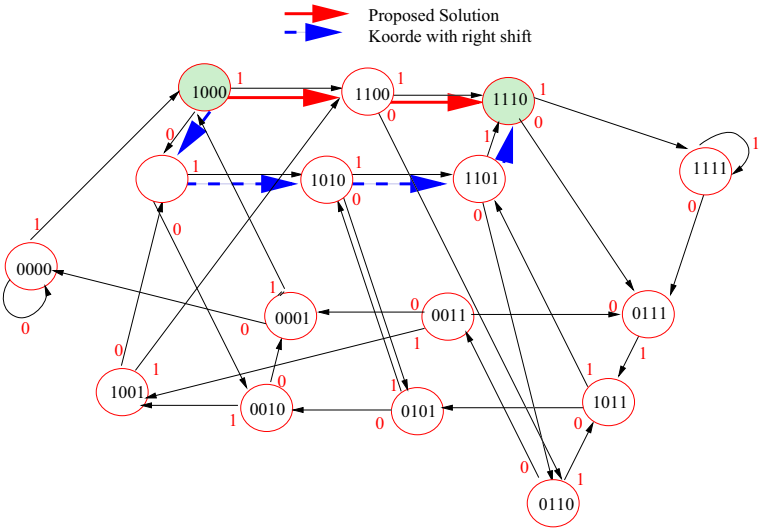

Fig. 4 Example of a routing process for both proposed and Koorde methods. (Color figure online)

In Koorde method, we use right shift operation instead of left shift, because in our graph topology, the construction is based on this operation. As shown in Fig. 4, routing path from node 1000 to node 1110 given by our proposed solution is with solid lines (*red color*). However, the given routing path from the same source to the same destination using right shift based Koorde method is illustrated with dashed lines (*blue color*).

3.5 Performance Evaluation

For validation purpose, we develop a specific simulation tool (*Java program*), and all tests are obtained on a PC platform with the following characteristics: 2.16 GHz and 1 GB of RAM.

In order to evaluate the performance of the proposed algorithm, we compare it with Koorde; one of the main existing solutions based on De Bruijn graph, in terms of cost lookup. Five tests are performed for three different diameters (5, 6 and 7) of the used De Bruijn graph. The obtained results are summarized in Figs. 5, 6 and 7. Test (i, j) indicates that the source is i and the destination is j.

Figures 5, 6 and 7 show the average number of hops corresponding to a lookup request for both Koorde and our proposed algorithm. for the three scenarios, the same source i and destination j are considered (*X-axis*) and the diameters of the considered networks is respectively 4, 5 and 6. The number of hops for both algorithms and for the three scenarios is less than the network diameter. However, the proposed algorithm shows better performances than Koorde, in terms of average number of hops. The average performance rate is equal to $\frac{(2+3+4+2+3)*100}{5*4} = 70\,\%$ for the scenario represented by Fig. 5, and it is equal to $\frac{(2+2+4+2+4)*100}{5+2+5+2+5} = 73.68\,\%$ for the scenario represented by Fig. 6, and it is equal to $\frac{(5+5+4+4+3)*100}{6+6+4+6+6} = 75\,\%$ for the scenario represented by Fig. 7.

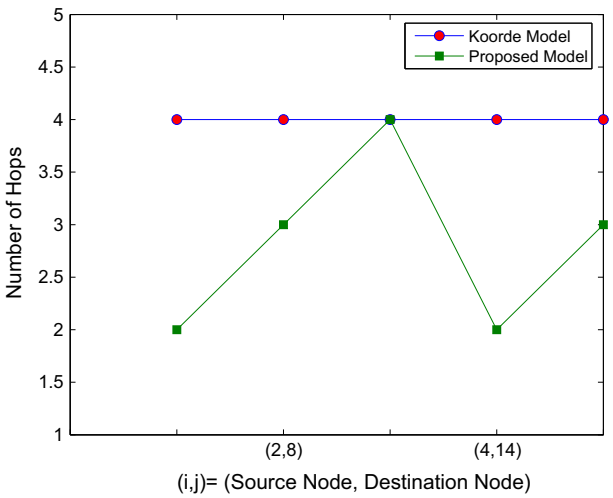

Fig. 5 Comparison with diameter equal to 4

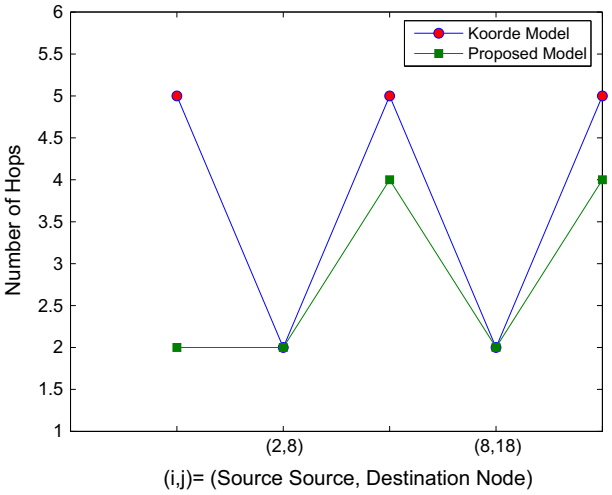

Fig. 6 Comparison with diameter equal to 5

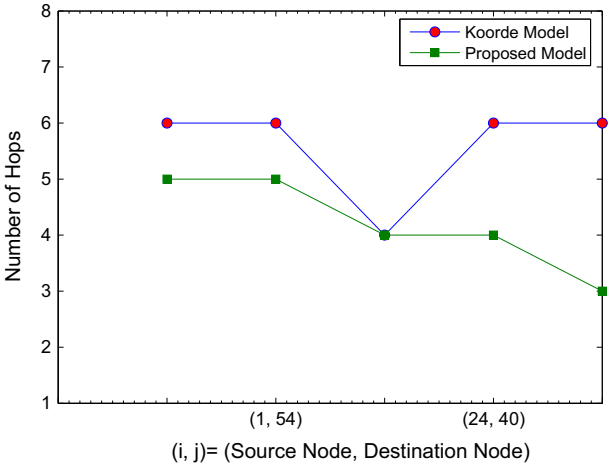

Fig. 7 Comparison with diameter equal to 6

4 Conclusion and Future Works

P2P lookup acceleration and optimization is still a challenge. Specific graphs (e.g. *De Bruijn graph*) have been exploited to accelerate and optimize routing process. However, The complexity of the network management and stabilization process is critical and should be limited.

In this paper, we have proposed a scalable solution for lookup acceleration and optimization based on De Bruijn graph with right shift. The proposed solution is principally based on the determination and elimination of the common string between source and

destination. This procedure is executed locally at the current requestor node, and then it is a rapid process.

The performance aspects of our proposed model have been validated thought simulation results. For this, a specific java program was developed. The results we obtained show that the proposed algorithm outperforms Koorde, in terms of cost lookup (*number of hops*).

In term of perspectives, we envision to measure the performances of our proposition in environment in highly dynamic environments, with high churn rate, while taking into consideration some physical QoS metrics such as end to end delay instead of the number of hops.

Acknowledgments The authors would like to thank Dr. B. Rabta from SOW-VU at Vrije University, Amsterdam for their valuable comments.

References

1. Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). *A scalable content addressable network*. New York: ACM SIGCOMM.
2. Shah, B., & Kim, K.-I. (2014). Towards enhanced searching architecture for unstructured peer-to-peer over mobile ad hoc networks. *Journal of Wireless Personal Communications*, *77*(2), 1167–1189.
3. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet application. In *Proceeding of the ACMSIGCOMM'01* (pp. 149–160). San Diego, CA, USA.
4. Rowstron, A. I. T., & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM international conference on distributed systems platforms heidelberg* (pp. 329–350). London, UK.
5. Amad, M., Meddahi, A., Aïssani, D., & Zhang, Z. (2012). HPM: A novel hierarchical peer-to-peer model for lookup acceleration with provision of physical proximity. *Journal of Network and Computer Applications*, *35*(6), 1818–1830.
6. Amad, M., & Meddahi, A. (2008). DV-Flood: An optimized flooding and clustering based approach for lookup acceleration in P2P networks. In *The international wireless communications and mobile computing conference, IWCMC08* (pp. 559–564). Greece.
7. Goodrich, M. T., Nelson, M. J., & Sun, J. Z. (2006). The rainbow skip graph: A fault-tolerant constant-degree P2P relay structure. In *Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithms, SODA 2006*. Miami, Florida, USA.
8. Shi, C. Q., Wang D. W., Huang H., & Huang Y. (2009). A clustering route algorithm of P2P networks based on Knödel graph. In *Proceeding of 2009 international conference on signal processing systems* (pp. 837–838), Singapore.
9. Ryu, J., Noel, E., & Tang, K. W. (2012). Distributed and fault-tolerant routing for borel cayley graphs. *International Journal of Distributed Sensor Networks, 2012*, Article ID 124245. doi:10.1155/2012/124245.
10. Yasuto, S., & Keiichi, K. (2003). An algorithm for node-disjoint paths in Pancake graphs. *IEICE Transactions on Information and Systems*, *E86–D*(12), 2588–2594.
11. Mandal, S., Chakraborty, S., & Karmakar, S. (2015). Distributed deterministic 1–2 skip list for peer-to-peer system. *Journal of Peer-to-Peer Networking and Applications*, *8*(1), 63–86.
12. Fraigniaud, P., & Gauron, P. (2006). D2B: A de Bruijn based content-addressable network. *Journal of Theoretical Computer Science*, *355*(1), 65–79.
13. Kaashoek, F., & Karger, D. R. (2003). Koorde: A simple degree-optimal hash table. In *Proceeding of the 2nd international workshop on peer-to-peer systems (IPTPS '03), LNCS 2735* (pp. 98–107).
14. Loguinov, D., Casas, J., & Wang, X. (2005). Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. *IEEE/ACM Transactions on Networking*, *13*(5), 395–406.
15. Spinsante, S., Andrenacci, S., & Gambi, E. (2011). Binary De Bruijn sequences for DS-CDMA systems: Analysis and results. *EURASIP Journal on Wireless Communications and Networking*,. doi:10.1186/1687-1499-2011-4.
16. Naor, M., & Wieder, U. (2003). Novel architectures for P2P applications: The continuous-discrete approach. In *Fifteenth ACM symposium on parallelism in algorithms and architectures (SPAA)* (pp. 50–59). New York, NY, USA.

17. Naor, M., & Wieder, U. (2007). Novel architectures for P2P applications: The continous-discrete approach. *ACM Transactions on Algorithms, 3*(3), 50–59.
18. El-Ansary, S., & Haridi, S. (2006). An overview of structured P2P overlay networks. In *Handbook on theoretical and algorithmic aspects of sensor, ad hoc wireless, and peer-to-peer networks*. ISBN: 978-0-8493-2832-9, Auerbach.

Mourad Amad received the engineer degree from the National Institute of Computer Science (INI-Algeria) in 2003 and the magister degree from the University of Bejaia (Algeria) in 2005. Currently, he is a Ph.D. at the University of Bejaia, Member of LaMOS Research Unit. His research interests include peer to peer networks (*architecture, application, security, VoIP*).

Djamil Aïssani started his career at the University of Constantine in 1978. He received his Ph.D. in 1983 from Kiev State University (Soviet Union). He is at the University of Bejaia since its opening in 1983/1984. Director of Research, Head of the Faculty of Science and Engineering Science (1999–2000). Director of the LAMOS research Unit (Modelling and Optimisation of Systems—http://www.lamos.org), Scientific Head of the Doctoral Computer School (since its opening in 2003), he has taught at several universities (Algiers, Annaba, Rouen, Dijon, Montpellier, Tizi Ouzou, Stif,…). He has published many papers on Markov chains, queueing systems, reliability theory, inventory, risk theory, performance evaluation and their applications in some industrial areas as electrical networks and computer systems. He was the president of the national Mathematical Committee (Algerian Ministry of Higher Education and Scientific Research—1995–2005).

Ahmed Meddahi is professor at Institut Mines Telecom/Telecom Lille in the Computer Science and Networks department. He obtained a master degree from University of Lille (France), a Ph.D. from University of Evry and "Institut National des Telecommunications", and "HDR" (*accreditation to supervise research*) from UPMC, University Pierre et Marie Curie (Paris 6). His main research interests are focused on IP signalling, "VoIP" performance, Qos and "context aware" management, with a focus on P2P architectures.

Makhlouf Benkerrou received the engineer degree from Bejaia university in 2010, his research interest is about lookup service and data management in P2P networks.

Farouk Amghar received the engineer degree from Bejaia university in 2010, his research interest is about lookup service and data management in P2P networks.