A formal framework for the performance analysis of P2P networks protocols [*]

Angelo Spognardi Roberto Di Pietro
Università di Roma "La Sapienza"
Dipartimento di Informatica
Via Salaria, 113, 00198-Roma, Italy
{spognardi,dipietro}@di.uniroma1.it

Abstract

In this paper we propose a formal framework based on the Markov Chains to prove the performance of P2P protocols. Despite the proposal of several protocols for P2P networks, sometimes there is a lack of a formal demonstration of their performance: experimental simulations are the most used method to evaluate their performance, such as the average length of a lookup. In this paper we introduce a versatile model for the analysis of P2P protocols. We employ this model to formally prove which is the average lookup length for two sample protocols: BaRT and Koorde. We verify the effectiveness of the proposed framework also via extensive simulations.

1. Introduction

Peer-to-peer protocols allow building networks of huge sizes, with thousand or million of users that cooperate between themselves. P2P networks can be scalable, resilient and efficient, usually without the requirement of centralized servers, in a completely distributed fashion. P2P protocols provide under-layers for a large variety of services, like network storage, content distribution, web caching, searching, indexing and more.
The basic operations performed by the peers of a network are join, leave and lookup. With a join a peer can introduce itself to the other peers of a network. With a leave the peer drops the connections with the network. When a peer has joined the network, it can start to communicate with the other peers and in particular might need to localize a resource shared in the network: the operation of lookup is used for this purpose. The lookup is often the most required operation of the network and then can seriously influence the performance experienced by the peers. Moreover, when the number of nodes is very big, the localization of the resources scattered in the network can be a challenge, especially in absence of a centralized index of the resources.
In literature several protocols have been proposed to localize the shared resources without introducing high overhead or bandwidth consumption. The lookup performance of these protocols are usually proved as a result of simulations or by formulas obtained from informal observations. Protocols simulations are fundamental to take in account the real environment in which protocols must operate, without heavy assumption or simplifications. But the only simulations is not enough to formally prove the properties and the features of protocols. While sometimes there is lack of a formal proof of protocols performance, it is also almost impossible to find a generic tool for formally evaluate operations like peer-to-peer lookup. Often, to compensate the shortage in formalization for the lookup performance, protocols are provided with a proof as for the diameter of the network. Focusing on the diameter only can hide some aspects of the protocol and can overlook some particular executions of the lookup that are very far from the average case. Moreover, it does not guarantee the real scalability of the whole system.

This paper provides a formal framework for the computation of the expected performance of P2P protocols. We instantiate this model to formally study the performance of lookup protocols in BaRT [16] and Koorde [9]. We focused on BaRT and Koorde to ease the exposition of the proposed methodology. One of our future works is to apply this methodology to Chord and Pastry as well. Note that the formal study of the average length of the lookup can be useful to determine the sensibility of every peer and the capacity of the whole peer-to-peer network. The proposed framework is based on the Markov Chains and is general enough to analyze a large variety of protocols. Besides, the use of one single tool makes also more easy and fair the comparison between different protocols.

[*]This work was partially funded by the *WEB-MINDS* project supported by the Italian MIUR under the FIRB program and by the European Commission in the framework of the "SatNEx" NoE project (contract N. 507052). Roberto Di Pietro is also with CNR-ISTI, WNLab-Pisa.

The remaining of the paper is structured as follow: next section makes a survey of the main P2P protocols and focus on the proof for their performance. Section 3 highlights the proposed methodology. In Section 4 and Section 5 we apply the proposed methodology to characterize the performance of lookup for the BaRT and Koorde protocols respectively. In Section 6, the theoretical results are compared with the experimental results obtained via simulations. Finally, in Section7 we provide some conclusive remarks.

2 Related Work

The popular P2P systems for file sharing, like Napster [12], Gnutella [7] and Freenet [3], do have a lack of formal proof for their performance.

The most popular P2P lookup protocol is Chord [17], which uses a distributed hash table to guarantee load balancing among peers, to perform lookups without the use of central servers, and to obtain a system with good resilience. The lookup performance of Chord are showed by simulations, while the theorem about the lookup length provides the bound of $\Theta(\log n)$. Other studies on the P2P protocol performance are proposed and use simulations or other empirical evaluation methodology [2], [14], [8] to justify their performance. Pastry [6] also proposes a decentralized object location for large-scale P2P system, but only simulations and informal discussions support the scalability and short lookup length. Other P2P systems like P-Grid [1] or H-Chord [4] are based on distributed hash tables and provide studies on the average case. Our analysis is focused on BaRT and Koorde protocol. The BaRT [16] protocol is based on a logical-tree view of the network: every peer is a leaf of a d-tree and the internal nodes of the tree are virtual. A protocol with some similarities with BaRT is Kademlia [11], but it is based on a Xor metrics and cannot be compared with the arbitrary degree of arity provided by BaRT. Koorde [9] can be view as an implementation of Chord over a network based on de Bruijn graph. From this kind of graph it inherits the optimal properties of constant node degree, connectivity and optimal diameter. As showed in [10], de Bruijn graphs have more and desirable properties and we remand to [10] for a more detailed analysis.
Finally, see [13] for a detailed introduction to Markov Chains and the calculation of the absorbing time.

3 Formal framework methodology

This section presents our formal methodology to formally prove the performance of P2P network protocols. The framework can have a wide range of applicability and in particular, to show its effectiveness, we will apply it to the analysis of lookup performance. Focusing on the lookup overhead is interesting, since the average lookup length can also be viewed as the overhead (in terms of bandwidth required or delay of the network) requested by the protocol to find out the required data. Moreover, the lookup operation is useful to calculate the "capacity" of the network, that is the average available throughput of each peer in the network: since a peer must forward an average of ℓ requests originated by other peers for every request performed, the expected useful capacity is the inverse, ℓ^{-1}.
Our methodology counts four steps:

1. protocol analysis;
2. identification of the "states";
3. extraction of the Transitions-matrix;
4. calculation of the average absorbing time.

A **protocol analysis** must be performed to understand the principles on which the protocol is based and to identify the actors and the actions of the protocol. The main goal is to depict a model of the protocol based on states. For instance, in the case of Koorde and BaRT, from the analysis we are able to recognize the peers as actor and the lookup as actions; then, we want to analyze the performance of the protocols w.r.t. the lookup length and, consequently, we use as metric the number of hops taken to perform a lookup. All the information collected during this phase will be useful to identify the states of the Markov Chain and the matrix of the transitions.

The **states identification** step requires to identify the states of the protocol that can be viewed as a state of a Markov Chain. Generally, we can say that a state is a snapshot of the evolution of the protocol in which the actors share the same probability to be the subjects or objects of an action (e.g. destination of a lookup, receiver of a message, subject to failure). For example, in Koorde we have introduced one state for each peer of the network: indeed, we can follow the evolution of a lookup as the sequence of peers that the lookup crosses to reach its destination. In BaRT, instead, we can partition the set of all the peer and define state i as the event "the lookup has reached a peer within the partition i": in such way, we can see the lookup as the sequence of the partitions traversed by the lookup.

In the **transitions matrix extraction** step, the transition-edges between the states are detected and the probabilities of each transition calculated. The transitions are related to the evolutions of the protocol, and then to the relations between the different actions of the actors. For example, to model BaRT we calculated the transition from state i to state j as the probability to randomly pick a peer from a set of a cardinality that depends from i and j (see Section 2). The probabilities associated to the transitions are collected and grouped in the transition-matrix of the chain.

In the last step (**average absorbing time calculation**), a study of the Markov Chain built in the previous steps is needed in order to identify the transient and the

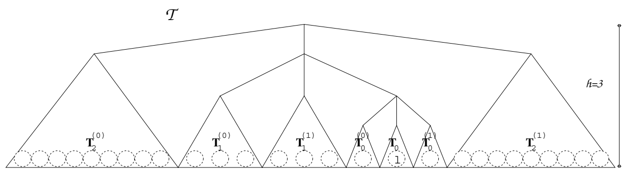


Figure 1. Tree as union of disjoint subtrees

recurrent states and, consequently, the absorbing classes. Since the model is related to properties of the protocol that we have to study, the average absorbing time of the chain will be the requested performance of the protocol.

4 The Balanced Randomized Tree protocol

This section presents the application of the proposed methodology to the BaRT protocol, while in Section 6 these formal results are compared with simulations. The BaRT protocol uses a Distribute Hash Table: every peer maintains a piece of the whole data structure that allows to route information within the network. The data structure is a tree where every non-leaf node has exactly $d \in \mathbb{N}^+$ children. Every peer is associated to a leaf of the tree. When we consider a peer associated to a leaf l of the tree, we can view that the whole tree $\mathcal{T}$ is composed by the union of $(d-1)(h-1)+1$ subtrees $T_0, T_0^{(0)}, T_0^{(1)}, \ldots, T_0^{(d-1)}, T_1^{(0)}, \ldots, T_{h-1}^{(d-1)}$: T_0 is the subtree composed by leaf l only, while every subtree $T_h^{(i)}$ is the i-th subtree of height h from the left that does not contain l (see Figure 1, where $d = 3$). For each subtree $T_h^{(i)}$, the peer l knows the address of (has a link to) one peer associated to a random leaf of $T_h^{(i)}$: the whole set of these links is called the *links table* of peer l. The set of the keys is partitioned and almost uniformly (see [16]) scattered between the peers, in such a way that every peer p knows exactly in which subtree $T_h^{(i)}$ is the peer that maintains a key k. Then, p starts a lookup searching for k, requesting to its link q for $T_h^{(i)}$ to continue the lookup. If q has key k, it answers to p; else, q determines in which subtree $T_{h'}^{(i')}$ is the key k and then asks to its link for subtree $T_{h'}^{(i')}$ to continue the lookup. This process is iterated until the peer t storing the key k is reached.

4.1 BaRT Analysis

Before presenting the model for the BaRT analysis, we depict a basic scenario to introduce the notation used in the following. Assume, without loss of generality, to have a balanced and complete binary tree $\mathcal{T}'$, of height h', with $N' \leq 2^{h'}$ leaves that represents the logical structure of a BaRT network. Consider $\mathcal{T}$, subtree of $\mathcal{T}'$, of height $h \in \mathbb{N}^+$ and with $N = 2^h$ leaves. $\mathcal{T}$ has N peers associated, one for every leaf of the tree and every peer has $\log N$ links toward peers in the tree $\mathcal{T}$ and $h' - h$ other links to the peers outside $\mathcal{T}$, according to the rules of BaRT. The tree $\mathcal{T}$ can be viewed as the union of the h disjoint subtree $T_0, T_1, T_2, \ldots, T_h$ of height respectively $0, 0, 1, \ldots, h-1$, as in Figure 2. A peer p of the network, associated to a leaf outside the tree $\mathcal{T}$ (see Figure 2), must perform a lookup for a key k stored in $\mathcal{T}$. The key k is stored by the peer associated to the (only) leaf q in the tree T_0.

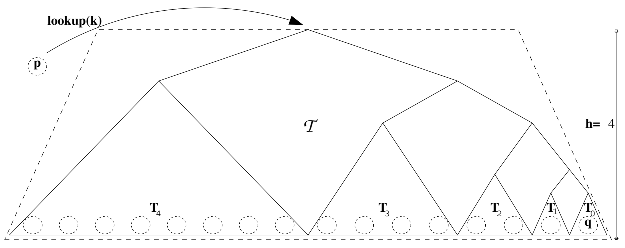


Figure 2. Lookup scenario

Now we are ready to introduce the model. We consider a discrete time homogeneous chain $\{X_t\}_t$, with $t \in T \subset \mathbb{N}^+$. The chain represents the succession of the trees that the lookup hits to reach q. Then, every X_t assumes a value in the set $E = \{0, 1, \ldots, h\}$, that is in the set of the indexes of the subtrees $(T_0, \ldots, T_h)$ that compose the tree $\mathcal{T}$. In particular, the random variable $X_n = i$ indicates subtree T_i as the subtree in which is the leaf that, at time n, must perform the next step of the lookup.

Due to the randomization of the links of a subtree if the lookup must be forwarded in a peer of T_i, every peer of T_i has the same probability p_i to be the peer reached by the lookup. Then, the following fact holds:

Fact 1. $p_i = \frac{1}{\#\text{ of leaves of } T_i}$

For the probability of a subtree T_i to be the next subtree hit by the lookup, it is not relevant which leaf r of a subtree T_i receives the request to perform the next step: it is only relevant the number of leaves of the subtree T_i.

Theorem 1. *The probability that the first leaf reached by the lookup lies in the subtree T_i is:*

$$P\{X_0 = i\} = \begin{cases} \frac{1}{2^h}, & \text{if } i = 0 \\ \frac{1}{2^{h-(i-1)}}, & \text{otherwise} \end{cases}$$

The proof for this theorem and for the followings ones are omitted due to space limitation and can be found in [15]. We can calculate the probability p_{ij} that the lookup will be forwarded from a subtree T_i to a subtree T_j. First of all note that, in absence of failures, the lookup process of the BaRT protocol is such that a peer requests the next step of a lookup only to a peer in a subtree of height smaller than its own. This means that the probability that the lookup goes back in a subtree already hit is 0, that is:

$$P\{X_n = j | X_{n-1} = i, j \geq i\} = 0$$

To request the next step of the lookup, a peer r in the tree T_i refers to its link of the subtree rooted at the brother of the root of T_i, the subtree $\widehat{T_i} = T_{i-1} \cup T_{i-2} \cup \ldots \cup T_0$. $\widehat{T_i}$ has the same height h_i of the subtree T_i and, then, the same number of peers. Thanks to the randomization, each of these peers can be the link of r to reach $\widehat{T_i}$; the peer than will perform the next step of the lookup. This shows that for tree $\widehat{T_i}$ also holds Fact 1.

Theorem 2. *If $0 < j < i \leq h$, then $p_{ij} = \frac{1}{2^{i-j}}$.*

Corollary 1. *For each $i, 0 < i \leq h$, $p_{i0} = p_i = \frac{1}{2^{i-1}}$*

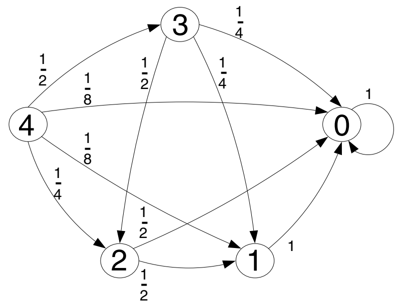
For example, with reference to Figure 2, assume that the lookup for key k stored by peer q has reached a peer r associated to a leaf of subtree T_3. Since the target of the lookup is the key k, r will forward the request to the subtree $T_2 \cup T_1 \cup T_0$, that has $2^{h_3} = 4$ leaves. Then, the next peer hit by the lookup will be a peer in T_2 with probability $\frac{1}{2}$, the peer in T_1 with probability $\frac{1}{4}$ and the peer in T_0 (i.e. q) with probability $\frac{1}{4}$.
When the lookup reaches q (that is subtree T_0), the lookup ends. This can be formalized with $p_{00} = 1$. Now, we are able to build the transition matrix P, using these elements:

$$p_{ij} = \begin{cases} 0, & \text{if } i \leq j \neq 0 \\ \frac{1}{2^{i-j}}, & \text{if } 0 < j < i \\ \frac{1}{2^{i-1}}, & \text{if } i \neq j = 0 \\ 1, & \text{if } i = j = 0 \end{cases}$$

obtaining a matrix like that in Figure 3. We can represent the chain as an oriented graph in which: the vertices are the states of the chain and between two vertex i and j there is an edge (i, j) labeled with p_{ij} if $p_{ij} > 0$. In Figure 3 we report the transition matrix and the corresponding graph for the chain of the tree $\mathcal{T}$ in Figure 2. Observing the matrix P, we can notice that all the states $1 \ldots h$ of the chain are transient except state 0, that is the only recurrent state. Moreover, state 0 is also the only absorbing state, and constitutes an irreducible closed class C. Then, calculating the average absorbing time τ_i of the class C, we calculate the average number of states to pass through, starting from state $i \in E$, before hitting a state of the closed class C. In other words, τ_i represents the average number of subtrees hit by a lookup that starts from T_i to reach the peer q, target of the lookup. Recall that τ_i (the set of average absorbing time starting from $i \in E$) can be calculated as the solutions of the linear system

$$\tau_i = 1 + \sum_{r \in T} p_{ir}\, \tau_r \tag{1}$$

where T is the set of the transient states of the chain. We can observe that P is a lower triangular matrix, and then the solutions to Equation (1) can be determined by recursion, starting from $\tau_0 = 1$ and $\tau_1 = 1$:



P	0	1	2	3	4
0	1	0	0	0	0
1	1	0	0	0	0
2	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
3	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0	0
4	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	0

Figure 3. Graph and transition matrix of a tree with $h = 4$

$$\begin{aligned} \tau_i &= 1 + p_{i\,1}\,\tau_1 + p_{i\,2}\,\tau_2 + \ldots + p_{i\,i-1}\,\tau_{i-1} = \\ &= 1 + \frac{1}{2^{i-1}}\,\tau_1 + \frac{1}{2^{i-2}}\,\tau_2 + \ldots + \frac{1}{2^{i-(i-1)}}\,\tau_{i-1} \end{aligned} \tag{2}$$

To eliminate the recursion and simplify the computation we prove the following theorem:

Theorem 3. *Let*

$$\tau_i = 1 + \frac{1}{2^{i-1}}\,\tau_1 + \frac{1}{2^{i-2}}\,\tau_2 + \ldots + \frac{1}{2^{i-(i-1)}}\,\tau_{i-1}$$

and $T_i = \frac{1}{2}(i+1)$, then $\tau_i = T_i \quad \forall\, i \in \mathbb{N}^+$

With the above theorem, we have that the value for τ_i, the average number of subtrees hit by a lookup within a binary tree of height i, is provided by the value of T_i. Since for the hypothesis the tree has height $h = \log n$, we obtain that the average lookup length of a lookup τ_h is equal to $\frac{1}{2}(\log n + 1)$.

Now, we can generalize to BaRT networks based on trees of arity $d \in \mathbb{N}^+$. The process is the same: fix arity d and key k for the lookup, we can suppose without loss of generality, that the tree $\mathcal{T}$ is complete with height h, and it is a subtree of a higher subtree $\mathcal{T}'$ that represents the logical structure of the BaRT network. The tree $\mathcal{T}$ can be viewed as the union of $(d-1)(h-1)+1$ subtrees $T_0, T_0^{(0)}, T_0^{(1)}, \ldots, T_0^{(d-1)}, T_1^{(0)}, \ldots, T_{h-1}^{(d-1)}$: T_0 is the subtree of height 0 that contains the peer q target of the lookup and $T_{h_i}^{(j)}$ is the j-th subtree of height h_i from the left. Denote with T_i the set of subtrees with the same height h_i. We can see the lookup process for the key k as the sequence of subtrees hit by the lookup from a peer p: every X_t assumes a value in the set $E = \{0, 1, \ldots, h\}$, that is in the set of the indexes of the set $(T_0, \ldots, T_h)$ that compose the tree $\mathcal{T}$. Each random variable $X_n = i$ indicates that at time n a peer in a subtree of T_i must

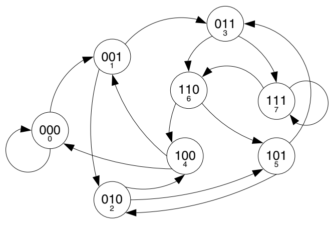


Figure 4. De Bruijn graph, with $d = 2, b = 3$

perform the next step of the lookup. Repeating the logical steps shown for binary trees, we can calculate the probability for every peer to be the link of p for tree $\mathcal{T}$ as $p_i = \frac{1}{d^{h-1}}$.

Moreover, generalizing Theorem 2, we have that:

$$p_{ij} = \begin{cases} 0, & \text{if } i \le j \ne 0 \\ \frac{d-1}{d^{i-j}}, & \text{if } 0 < j < i \\ \frac{1}{d^{i-1}}, & \text{if } i \ne j = 0 \\ 1, & \text{if } i = j = 0 \end{cases}$$

and obtain the required matrix P. Solving Equation (1), we obtain the recursive formula:

$$\tau_i^{(d)} = 1 + \frac{d-1}{d^{i-1}}\,\tau_1^{(d)} + \frac{d-1}{d^{i-2}}\,\tau_2^{(d)} + \ldots + \frac{d-1}{d^{i-(i-1)}}\,\tau_{i-1}^{(d)} \quad (3)$$

Using a straightforward generalization of Theorem 3, we can show that, fixed $i = h$:

$$\tau_h^{(d)} = T_h^{(d)} = \frac{1}{d}\big((d-1)\,h + 1\big) \quad (4)$$

that is, the average number of subtree hit by a lookup to reach a target peer effectively depends on the height of the tree. If the tree is balanced the number of exchanged messages between peers is logarithmic in the number of peers of the network, where the base of the logarithm is d. With Equation (4), we can prove another result for BaRT; set $d = \log n$, then with a tree where $h = \log_d n$ we obtain that

$$T_h^{(d)} = \frac{\log n - 1}{\log\log n} + \frac{1}{\log n} \quad (5)$$

that is, the length of a lookup is $\Theta\left(\frac{\log n}{\log\log n}\right)$.

5 The Koorde protocol

The Koorde [9] protocol is a Distributed Hash Table based on Chord [17] and the de Bruijn graphs [5]. The main characteristic of this kind of graphs is the constant degree d of nodes. This model is completely characterized by the graph described as (V, E), where $V = \{0, \ldots, n-1\}$ $(n = d^b - 1, \text{with } b \in \mathbb{N})$ and $\forall u \in V$, $\exists (u, v_1), \ldots, (u, v_d) \in E$, where $v_i = d * u + (i-1) \bmod d^b$ (see Figure 4 for a pictorial representation of an instance of such a model). Informally, the set of vertices is the set of the first d^b natural numbers and every vertex has exactly d outgoing edges, one toward every vertex with the identifier between $d*v$ and $(d+1)*v-1$, where all the operations are performed modulo d^b.

Without loss of generality, we can instantiate a Koorde network where the degree of the de Bruijn graph is $d = 2$, the number of nodes is $n = 2^m, m \in \mathbb{N}$, and also the number of keys are n, ordered from 0 to $2^m - 1$. Every peer of the Koorde network is mapped on a vertex of the graph: the peer mapped to vertex p is labelled with the binary representation of value p, referred as $\hat{p}$ and stores the key p. Every key k is labelled with the binary representation of k, that is $\hat{k}$. Every peer has two links, according to the outgoing edges of its associated vertex. To perform a lookup of k, a peer with identifier p finds the longest sequence of bits in which the less significant bits of $\hat{p}$ and the most significant bits of $\hat{k}$ match. Then, p shifts $\hat{p}$ to the left, in according to the bit of $\hat{k}$ in which the sequence differs, obtaining the value $t = \hat{p}_l \circ topBit(\hat{k})$ ($\circ$ denotes the operation of concatenation and $\hat{p}_l$ is the sequence $\hat{p}$ without the first bit). Then p forwards the lookup to the node with identifier t, which is one of its neighbours. The process is iterated until the request reaches the peer that maintains the key k.

The same steps have to be performed for a lookup also if the degree d of the graph is greater than 2: it is sufficient to take in account a bigger set of symbols Σ to encode every identifier of keys and nodes [10].

5.1 Koorde Analysis

From now on, we suppose that the network in analysis has built on a de Bruijn graph with degree d and that the number of peers of the network is $n = d^b - 1$. To analyze the model presented above, we can consider a typical scenario: we have a random peer p that has to perform a lookup for a random key k, where $p, k \in V = \{0, \ldots, n\}$.

We can divide the space of identifiers V in $b+1$ subsets $C_0, C_1, \ldots, C_b$, where:

$$\begin{cases} C_0 & = \{p\} \\ C_i & = \{dx + j \bmod d^b, j = 0, \ldots, d-1 \bmod d^b \\ & \qquad\qquad\qquad\qquad \forall x \in C_{i-1} \setminus \ C_{i-2}\} \end{cases}$$

that is the set of nodes reachable from p the first time respectively in $0, 1, \ldots, b$ steps. Notice that every set C_i contains all the peers that have the first $b - i$ most significant bits of the identifier equal to the $b - i$ less significant bits of the identifier of p.

If we consider the Markov Chain X_n that represents the set in which the lookup for key k is at step n, we can build the corresponding matrix:

P	0	1	2	3	...	z	z+1	...	b
0	0	**1**	0	0	...	0	0	...	0
1	0	0	**1**	0	...	0	0	...	0
2	0	0	0	**1**	...	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
z-1	0	0	0	0	...	**1**	0	...	0
z	0	0	0	0	...	**1**	0	...	0
z+1	0	0	0	0	...	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
b	0	0	0	0	...	0	0	...	0

Notice that the state z corresponds to the set C_z and contains the peer that maintains key k. Indeed, after z shift the identifier of p becomes the identifier of k. The matrix P identifies state z as the only recurrent state and also the only absorbing state. Then, fixed z, using Equation 1 we can calculate the average absorbing time of the chain, starting from C_0. For every τ_i, the only non-zero value of Equation 1 is $p_{i\ i+1}\tau_{i+1}$. Then, Equation (1) corresponds to:

$$\tau_i = 1 + \sum_{j=i+1}^{z-1} \tau_j$$

except for $\tau_z = 1$, where we obtain recursively that $\tau_{z-i} = i$ and, finally:

$$\tau_i = z - i.$$

For $i = 0$ (the state that starts the lookup) we have that $\tau_0 = z$, that is, fixed z, the number of steps that a lookup must perform in average to hit the target is z. To evaluate the average length of a lookup, we use the function $f(i)$ defined in [10] that approximates the distribution of the lookup length, that is:

$$p(i) \approx f(i) = \frac{d^i}{n}\left(1 - \frac{d^i + d^{i-1} - 1}{dn}\right), n \geq 1$$

Then we obtain that

$$E[\tau_d] = \sum_{i=0}^{b} i\ p(i) \approx \sum_{i=0}^{b} i\ \frac{d^i}{n}\left(1 - \frac{d^i + d^{i-1} - 1}{dn}\right)$$

6 Experiments and Discussion

In this section we report the simulation results for the BaRT and Koorde protocols and compare these experimental results with the analytical results from the previous sections. In Koorde, we randomized the peer requesting a lookup query as well as the requested key. We counted the number of hops needed for the lookup to reach the node that stores the requested key. In BaRT, an experiment consists of two phases: generation of the network and lookup execution. During the network generation, every peer of the network receives its random links table, according to BaRT rules. To perform the test of a single lookup, we randomly chose a requesting peer and a key and counted the number of peers involved by the lookup. Every test was repeated 25 times and the reported results are the averaged values.

BaRT networks for arity 2. This section presents the data obtained by simulating BaRT using binary trees. The graph in Figure 5 compares the lookup lengths for networks of different sizes. The number of nodes n of a network is doubled starting from 128 up to 8192. The graph plots also the function $T_2^{(h)}$ of the Theorem 3,

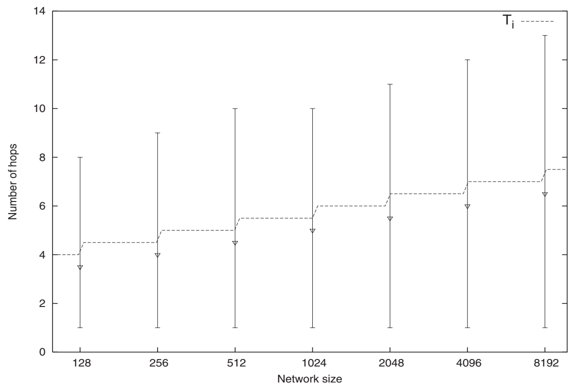


Figure 5. BaRT: Min/med/max experienced values for lookup length varying n, d=2

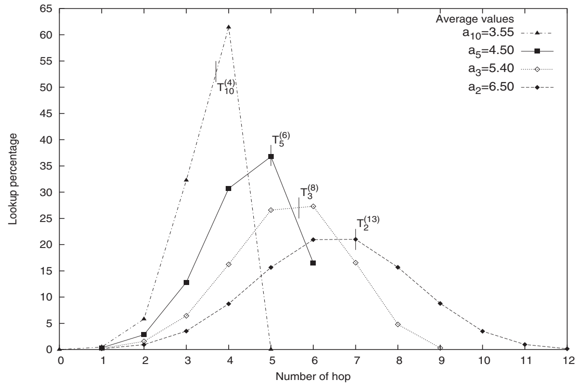


Figure 6. BaRT: lookup length varying d

to compare the simulated values with the expected values. We assumed tree of height $\lceil \log n \rceil$. The bars of the graph report the minimum and the maximum lengths of a lookup, whereas the middle signs are the experimented average lengths of the lookups. The simulations confirm our theoretical analysis, because the values of the function $T_2^{(h)}$ $(i \in \mathbb{N})$ are very close to the average length values experimented.

BaRT networks for arity 3, 5 and 10. In Figure 6 are reported the comparison between results obtained simulating networks with a fixed size (8192 nodes), but varying the arity. In the plot the theoretical values are also reported for the expected average length for arity 2, 3, 5 and 10. To obtain the theoretical values, we calculated the succession of Equation (4) using for the height of the tree h the nearest value that can be expressed as a power of the arity. In particular, for the arity 3 we used a height of 8 because $8192 \approx 3^{8.2}$, while for arity 5 we used $h = 6$ because $5^5 = 3125 < 8192 < 15625 = 5^6$ (then we used respectively $\tau_3^{(8)}$ and $\tau_5^{(6)}$). For arity 10, we used $\tau_{10}^{(4)}$. Notice that, if we do not evaluate $T_i^{(d)}$

Table 1. BaRT

d	h	n,Size	$T_h^{(d)}$	$\overline{T}_i^{(d)}$	a_d, Exper. value
2	13	8192	7	7	6.50
3	8	8192	5.66	5.38	5.40
5	6	8192	5	4.32	4.50
10	4	8192	3.7	3.62	3.55
10	4	10000	3.7	3.7	3.61
10	4	50000	3.7	4.33	4.3
10	5	100000	4.6	4.6	4.6

Table 2. Koorde

d	n, Size	$b = \log_d n$	$E[\tau]$	a_d, Exper. value
2	8192	13	11.335	11.3378
3	6561	8	7.31319	7.31785
5	15625	6	5.69803	5.69881
10	10000	4	3.87772	3.87791
10	100000	5	4.87767	4.87771

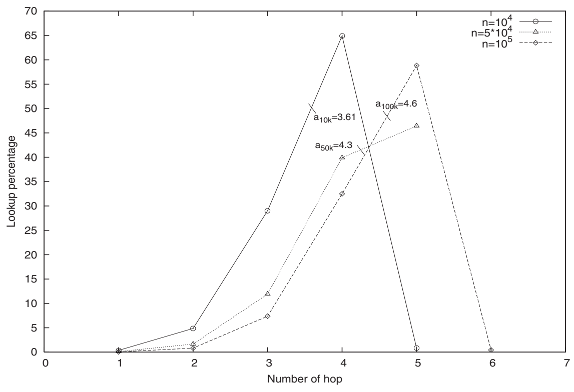


Figure 7. BaRT: lookup length varying n, d=10

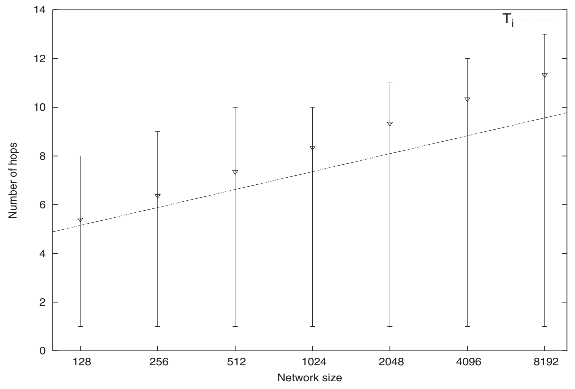


Figure 8. Koorde: Min/med/max experienced values for lookup length varying n, d=2

rounding to the nearest integer value, but use instead the $\log n$ function, we obtain more accurate theoretical values. All the values are reported in Table 1, where $\log_d(n) = i$. The not rounded values ($\overline{T}_i^{(d)}$) are reported as well. Figure 7 reports the results of three simulations performed with arity 10. The size of the network was of 10,000, 50,000 and 100,000 nodes, while 300,000 lookup were performed for every network size. For every plot, we reported the expected value ($T_{10}^{(4)}$ for 10,000 and 50,000, $T_{10}^{(5)}$ for 100,000) and the experienced average length (respectively a_{10k}, a_{50k} and a_{100k}).

Koorde networks, based on binary graph. In Figure 8 and Figure 9 we reported the results of the simulations for the Koorde networks with degree 2. Figure 9, in particular, shows the comparison between the simulated and the theoretical length of a lookup. The simulation is based on a network with 8192 peers ($n = 2^{13}$) in which we simulated 500,000 lookups, randomly selecting the requesting peer and the requested key. We reported in the graphic, for every lookup length k, the fraction of lookups that was performed in k steps. To show the theoretical value, we plotted the function $f(i)$, used in Section 5.1 to evaluate the average lookup length $E[\tau]$, and also reported the average value a_2 obtained from simulations. Considering only two decimal digits, we obtained the same value for $E[\tau]$ and a_2, that is $E[\tau] = a_2 = 11.33$. In fact, as we can see in the inner graphic (a magnification for the interval $[11 - 13]$) the differences between the different values are negligible.

Koorde networks based on non-binary graph. Figure 10 shows both simulated and theoretical values for the length of every lookup. The figure allows to make a comparison between different sizes of the network and different degrees. For all degrees we can observe a strict agreement between experiments and theoretical values. Figure 11 reports the values obtained simulating networks with degree $d = 10$ and with 10,000 and 100,000 nodes respectively. In Table 2 we summarize the values related to Figure 10 and Figure 11.

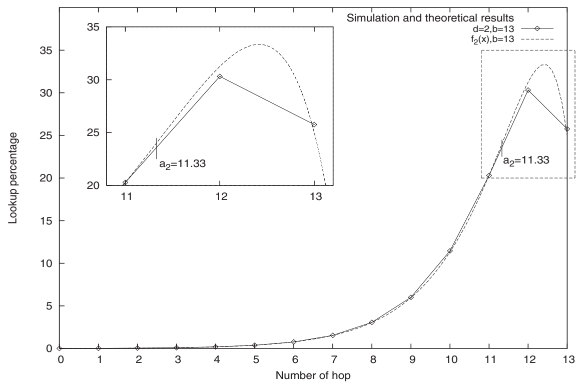


Figure 9. Koorde: lookup length, d=2, n=8192

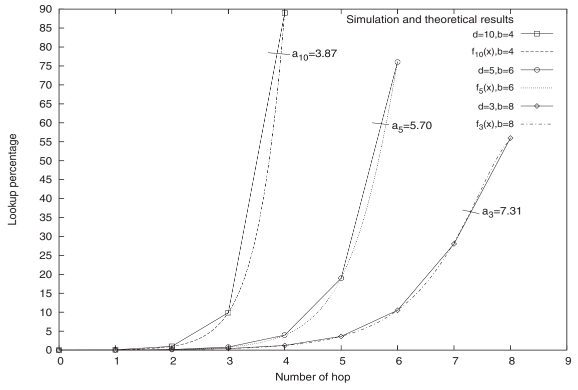


Figure 10. Koorde: lookup length varying d

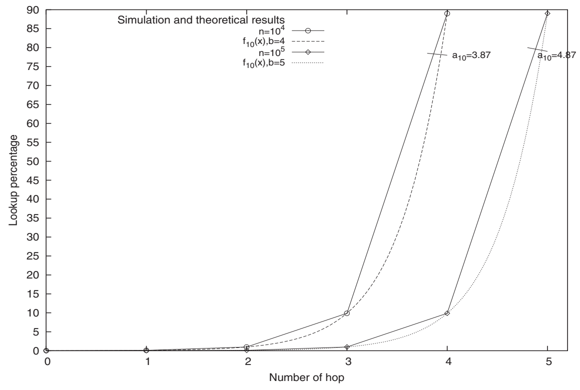


Figure 11. Koorde: lookup length, d=10

7 Conclusion

This paper described a methodology to assess the performance of a P2P network. The methodology is formal and based on Markov Chains. We have shown its validity focusing on the performance evaluation of lookup operations. In particular, we have employed the methodology to analyse two protocols: BaRT and Koorde and their relative performance as for the lookup operations. Extensive simulations confirmed the sharp validity of analytical results. Among future works, we envisage to adapt the methodology to Chord and Pastry as well; moreover, we are planning to extend the methodology to the analysis of other properties of interest in P2P network, such as connectivity.

Acknowledgments.
The authors are grateful to professor Luigi V. Mancini and the anonymous reviewers for providing useful suggestions and comments.

References

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *Proceedings of the* 9th *International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy*, 2001.

[2] J. Chu, K. Labonte, G. Bissias, D. LaFlamme, and B. N. Levine. A trace-driven evaluation of chord. Technical Report 04-38, Dept. of Computer Science, Univ. of Massachusetts, Amherst, MA 1003, 2004.

[3] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability. LNCS 2009. Springer Verlag.*, 2001.

[4] G. Cordasco, L. Gargano, M. Hammar, and V. Scarano. Degree-optimal deterministic routing for p2p systems. In *PODC '04*, New York, NY, USA, 2004. ACM Press.

[5] N. de Bruijn. A combinatorial problem. In *Proceedings of Koninklijke Nederlandese Akademie van Wetenschappen*, 1946.

[6] P. Druschel and A. Rowstron. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the* 18th *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.

[7] Clip2. *The Gnutella Protocol Specification v.0.4 (Document revision 1.2).*

[8] J. Guillaume, M. Latapy, and S. Le-Blond. Statistical analysis of a p2p query graph based on degrees and their time-evolution. In *LNCS Proceedings of the* 6th *International Workshop on Distributed Computing (IWDC'04) (Calcutta, Inde)*, volume 3326, 2004.

[9] D. Karger and M. F. Kaashoek. Koorde, a simple degree-optimal hash table. In *Proceedings of the* 2nd *International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, 2003.

[10] D. Loguinov, J. Casas, and X. Wang. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. *IEEE/ACM TON*, 13(5), 2005.

[11] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of IPTPS '02, Cambridge, MA, USA*, 2002.

[12] Napster web address: http://www.napster.com.

[13] S. M. Ross. *Probability Models*. Academic Press, 8th edition, 2002.

[14] S. Saroiu, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems, 2002. In Proceedings of Multimedia Conferencing and Networking (San Jose, CA), 2002.

[15] A. Spognardi and R. Di Pietro. A formal framework for the performance analysis of P2P networks protocols, TR-WEBMINDS-64. Technical report, Web-Minds, Unit of Rome, Jan. 2006.

[16] A. Spognardi, R. Di Pietro, and L. V. Mancini. Bart, balanced randomized tree: A scalable and distributed protocol for lookup in peer-to-peer networks. In *IEEE HOT-P2P'04*, pages 22–29. IEEE Press, 2004.

[17] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM TON*, 11(1), 2003.