Universidad Rey Juan Carlos
Departamento de Sistemas Telemáticos y Computación

Doctoral Thesis

Resource Location Mechanisms for Complex Networks Based on Random Walks

Víctor Manuel López Millán
vmlm@gsyc.es

Supervisors:

Antonio Fernández Anta
anto@gsyc.es

Luis López Fernández
llopez@gsyc.es

Madrid, July 2013

II

I authorize the defense of the Doctoral Thesis *Resource Location Mechanisms for Complex Networks Based on Random Walks* written by Víctor Manuel López Millán.

Thesis Supervisor

Antonio Fernández Anta
Fuenlabrada (Madrid), July , 2013

I authorize the defense of the Doctoral Thesis *Resource Location Mechanisms for Complex Networks Based on Random Walks* written by Víctor Manuel López Millán.

Thesis Supervisor

Luis López Fernández
Fuenlabrada (Madrid), July , 2013

DOCTORAL THESIS: *Resource Location Mechanisms for Complex Networks Based on Random Walks*

AUTHOR: Víctor Manuel López Millán

SUPERVISORS: Antonio Fernández Anta
Luis López Fernández

The committee named to evaluate the Thesis above indicated, made up of the following doctors:

PRESIDENT:

MEMBERS:

SECRETARY:

has decided to grant the qualification of:

Fuenlabrada (Madrid), , 2013

The secretary of the committee.

Víctor Manuel López Millán
Departamento de Sistemas Telemáticos y Computación
Universidad Rey Juan Carlos,
Fuenlabrada 28933 Madrid

To my dear wife Belén, for your support, understanding, patience, company, and love, since the beginning, hopefully for long years to come. In the hope that you can forgive those things I did not get right.

Acknowledgments

The personal project of writing a doctoral thesis has been in my mind for many years. It all started when my wife encouraged me to pursue a long standing dream, joining the University. The occasion showed up, and after careful consideration and with her unconditional support, I decided to abandon industry for a new life. Even then, the task of writing a thesis appeared so daunting in my mind that I felt I would never be able to fulfill it. It was not until I joined the research group of my advisors, Antonio Fernández Anta and Luis López Fernández, that I started to believe. With them I have learned that we must always quest for world class excellence. They have always supported me, both academically and personally. My most sincere admiration and gratitude to them, for their inspiration and for their patience when the progress was not as fast as we all desired. All my gratitude also goes to Vicent Cholvi, from the Universitat Jaume I, for his continuous help and encouragement, for which I am deeply in debt. I would also like to make a special mention of my colleagues at Universidad CEU San Pablo, for their support and trust along this time.

Regarding my family at home, they only too well know the long hours I have had to regretfully sneak from them. The generosity of my wife seem to have no limits. I have no doubt that without her support this dissertation could not have finished successfully. Our children Cristina, María and Alejandro will be happy to know that I have already finished my "doctorado", something that they do not understand too well, but that they know it will make their dad less busy. I thank them all for their love and for sharing their joy of life. And thanks to my parents and the rest of the family for always being there, with a special dedication to the memory of my father. I wish to thank my friends too, with my apologies for having almost disappeared for a long time.

Thanks to you all for helping me to reach this point, which is only one more beginning.

Abstract

Random walks have been proven useful in several applications in networks. For instance, they can be used to locate resources in complex networks. The networks considered in this dissertation are built randomly, and their topology is unknown, making the use of random walks very suitable. However, search lengths (i.e., the length of the random walks) can be large, and therefore some variants have been devised pursuing a trade-off between better performance and limited cost. A self-avoiding random walk, for instance, is one that tries not to revisit nodes, therefore covering the network faster. Its performance when locating resources has been analyzed using essentially empirical studies, since a strict analytical approach is hard, because it is not a Markovian stochastic process. In this dissertation, we propose an analytical model that allows to estimate the expected search length of self-avoiding random walks in networks with resource multiplicity. The model includes the possible availability of one-hop resource replication (i.e., nodes know about their neighbors' resources).

To further reduce search lengths, we propose a resource location mechanism based on building random walks by connecting together partial random walks previously computed at each network node. Resources found in each partial walk are registered, so that searches can jump over partial walks where the resource is not located. We assume that perfect recording of resources may be costly, and hence compact probabilistic data structures, like Bloom filters, are used at the cost of introducing false positives. This mechanism can be applied to networks with static or dynamic resources. In the latter case, resource information deteriorates as resources appear and disappear over time. Different procedures to select a partial walk from a node result in variations of this mechanism. In addition, partial walks can be either simple random walks or self-avoiding random walks.

Analytical models are provided to predict expected search lengths and other magnitudes of the resulting mechanisms. Simulation experiments performed on several types of networks characterized by their size and degree distribution are used to validate these predictions and to compare these tech-

niques with simple random walk searches. The experiments show very large reductions of expected search lengths, even in the face of high resource volatility.

Resumen

Los caminos aleatorios han demostrado ser útiles en diversas aplicaciones en redes. Por ejemplo, pueden utilizarse para buscar recursos en redes complejas. En esta tesis, consideramos redes construidas aleatoriamente y con topología desconocida. siendo adecuado el uso de caminos aleatorios. Sin embargo, las longitudes de las búsquedas que usan caminos aleatorios pueden ser muy grandes, por lo que se han diseñado variantes en busca de un compromiso entre rendimiento y coste. Un camino aleatorio *self-avoiding*, por ejemplo, intenta no revisitar nodos, cubriendo la red más rápidamente. Su rendimiento en la búsqueda de recursos se ha analizado de forma esencialmente experimental, ya que un estudio analítico estricto es difícil, debido a que no es un proceso estocástico de Markov como los caminos aleatorios normales. En esta tesis, proponemos un modelo analítico que permite estimar la longitud esperada de las búsquedas que utilizan caminos *self-avoiding* en redes con multiplicidad de recursos. El modelo incluye la posible existencia de replicación de recursos a un salto (es decir, los nodos conocen los recursos de sus vecinos).

Para reducir aún más las longitudes de las búsquedas, proponemos un mecanismo de construcción de caminos aleatorios que conecta entre sí caminos aleatorios parciales previamente realizados en cada nodo. Los recursos encontrados en cada camino se registran para que las búsquedas puedan saltar sobre caminos que no poseen el recurso. Asumimos que el almacenamiento de los recursos puede ser excesivo, por lo que se utilizan estructuras de datos probabilísticas compactas, como por ejemplo filtros de Bloom, con el coste de introducir falsos positivos. Este mecanismo puede ser aplicado a redes con recursos estáticos o dinámicos. En este último caso, la información registrada sobre los recursos se deteriora con el paso del tiempo. Diferentes procedimientos de selección de un camino parcial en un nodo dan lugar a distintas variantes de este mecanismo. Además, los caminos parciales pueden ser caminos aleatorios normales o caminos *self-avoiding*.

Para todos estos mecanismos de búsqueda, Proporcionamos modelos analíticos que predicen la longitud esperada de la búsqueda y otras magnitudes de

los mecanismos resultantes. Se utilizan experimentos de simulación en distintos tipos de redes caracterizadas por su tamaño y distribución de grado para validar estas predicciones y para comparar estas técnicas con los caminos aleatorios normales. Los experimentos muestran grandes reducciones de la longitud esperada de las búsquedas, incluso en presencia de una alta volatilidad de recursos.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the framework of the research performed for this dissertation. First, resource location in complex networks, a part of the distributed computing field, is established as context of the dissertation in Section 1.1. Then, Section 1.2 introduces random walks, a central concept in this research. The hypothesis of this dissertation is stated in Section 1.3, followed by an outline of our contributions to the field in Section 1.4. Finally, Section 1.5 presents the organization of the rest of this dissertation.

1.1. Resource Location in Complex Networks

Networks are the base of communication technologies, but they can also be found in many other fields including physics, biology, social sciences, epidemiology, etc. Many basic operations like construction, characterization, sampling, searching, etc., have been applied to networks, often as part of more sophisticated applications. The work presented in this dissertation is focused on the resource location problem in networks. The *resource location* problem is defined as finding a network node that holds a desired resource (a file, service, device, identifier, etc.). Search mechanisms use algorithms to navigate the network looking for the resource, starting from some source node. Search algorithms take network information as input to guide the search. In general, the more information available as input, the better performance of the search, at the expense of gathering and managing the required information. The performance of a search algorithm can be measured with parameters like the search time, the search length (number of hops to find the resource) or the bandwidth used. Centralized approaches rely on a central directory that keeps registers of resources and their location in the network. Distributed approaches avoid having a central server using different

strategies, like imposing a certain structure on the topology of the network, which helps to guide searches. Search algorithms can be deterministic or randomized.

This dissertation aims at designing and evaluating efficient algorithms that locate resources in a network using as little network information as possible, seeking a balance between the performance and the cost of obtaining and keeping up to date that information. In particular, it is proposed to use *random walks*, which have proved to be useful in networks about which we have little or no information (frequently due to their large size, lacking of defined topology and building mechanisms, and/or rapid changes). This is often the case with real networks, which display some unique features that do not show up in networks studied by the traditional graph theory. Such networks are called *complex networks*. A simple random walk can be used to search a network using no information other than that required to choose a neighbor (uniformly at random) to hop to if the current node does not hold the desired resource. The downside is that the resulting average search lengths are usually large.

In this dissertation, we describe the use of three strategies based on random walks to lower search lengths using additional information. In the first one, we will simply require the walk to avoid revisiting nodes whenever possible, since already visited nodes are known not to hold the resource. In the second strategy, we construct short random walks in advance, registering the resources held by their nodes. When trying to locate a resource, the walk will be able to skip the short walks known not to hold the resource, traversing only a short walk when the resource is known to be found. This is expected to save a significant number of hops in the search. In our third strategy, we deal with the dynamics of the network. In particular, resources appear and disappear randomly, causing the information registered by the short walks to go stale, which will make the searching walk to take suboptimal decisions. Several variations of these base mechanisms are defined in this dissertation. We have developed analytical models for them to facilitate the assessing of their performance. These analytical models have been validated using simulation experiments executed in several types of randomly built networks. Experimental results have also helped in drawing conclusions about the performance of the proposed resource location mechanisms.

1.2. Random Walks

A *random walk* in a network is a routing mechanism that chooses the next node to visit uniformly at random among the neighbors of the current

node. Random walks have been extensively studied in mathematics, where they have been modeled as finite Markov chains [39, 55, 66], and have been used in a wide range of applications such as statistic physics, population dynamics, bioinformatics, etc. When applied to communication networks, random walks have had a profound impact on algorithms and complexity theory. Although naive, random walks have been proved useful, especially in situations where there is no complete knowledge about the network or where the network changes frequently (like the Internet, the WWW, peer-to-peer (P2P) networks and wireless ad-hoc networks), as is referenced below. Some of the advantages of random walks are their simplicity, their small processing power consumption at the nodes, and the fact that they need only local information, avoiding the communication overhead necessary in other routing mechanisms. Random walks have been proposed as a base mechanism for multiple network applications, including network sampling, network searching, network construction, and network characterization [2, 3, 10, 16, 25, 31, 32, 50, 51, 57, 56, 58, 78, 80, 85]. They have also been proposed as a solution for the resource location problem. Roughly speaking, this problem consists of finding a node that holds the resource, starting at some *source node*. Random walks can be used to perform such a search as follows. The source node first checks if it holds the desired resource. If it does not, it chooses one of its neighbors uniformly at random and sends a message to it, indicating the desired resource. In other words, the search *hops* to that neighbor. This node repeats the process, checking for the resource and forwarding the message. The search proceeds through the network in this way until a target node, i.e., one that holds the desired resource, is visited. Then, the target node sends a notification message to the source node. Due to the random nature of the walk, some nodes may be visited more than once (unnecessarily from the search standpoint), while other nodes may remain unvisited for a long time. The number of hops taken to find the resource is called the *search length* of the walk. The performance of this direct application of random walks to network search has been studied in [2, 32, 56, 78, 90].

The use of random walks for resource location has several clear applications, like content-centric networks (CCN) [40] or unstructured peer-to-peer (P2P) file-sharing systems. The former are networks in which the key elements are named content chunks, which are requested by users using the content name. Content chunks have to be efficiently located and transferred to be consumed by the user. Application of random walks to the latter systems is extensively described in Chapter 2.

1.3. Hypothesis

Context: We consider randomly built networks whose nodes hold resources. The nodes of the network have no information on the network topology other than the nodes directly connected to them. Nodes are interested in finding desired resources located at some node or nodes in the network. Searches based on simple random walks can be applied to locate those resources.

Hypothesis: Variations of random walks or combinations with other techniques can be used to devise efficient resource location mechanisms that outperform simple random walk searches in terms of the average search length.

1.4. Contributions

This section summarizes the contributions of this dissertation. In a nutshell, this work consists of the definition of several resource location mechanisms, and the corresponding performance evaluation studies, which are achieved using both an analytical approach and an experimental approach. Analytical models have been developed for each of the devised algorithms. Then, experiments have been performed to validate the predictions from the analytical models. Both analytical and experimental results are used to draw conclusions on the performance of the algorithms.

The main ideas behind the proposed mechanisms are:

- The use of self-avoiding random walks instead of simple random walks.
- The use of short precomputed partial walks to efficiently build longer random walks.
- The combination of partial walks and self avoiding walks.
- The use of deterministic or probabilistic information storage for the searches.
- The definition of variations of the proposed algorithms, derived from their particular features.

The performance evaluation of the mechanisms has been studied in different types of networks and with different ways of assigning resources to nodes:

- Several sizes and node degree distributions.

- The availability or unavailability of one-hop resource replication in the network (i.e., nodes know about their neighbor's resources).
- The existence of a single instance or multiple instances of the resource that is looked for in the network.
- The static or dynamic behavior of resources in the network.
- Several parametrizations of each of the algorithms, according to their particular features.

The following sections briefly present these ideas, which are fully described in Chapters 4 to 6.

1.4.1. Self-Avoiding Walks

One possibility to improve the performance of simple random walks is to slightly change the algorithm to try not to revisit nodes that have been already visited by the walk, clearly increasing the chance to visit the node that holds the desired resource. We refer to this variation of a random walk as a Self-Avoiding Walk (SAW). SAWs will be defined in detail in Chapter 3 in the context of this dissertation.

This dissertation studies SAW as a search mechanism in communication networks. The main contribution is an analytical model that estimates the average search performance of SAWs in complex networks with or without one-hop replication, and where a number of instances of the resource sought are present. In a *one-hop replication* network, a node knows about the resources held by its neighbors. Therefore, to find a resource in such a network it suffices to visit a node that holds it *or* any of its neighbors, whereas the walk must visit the node that holds the resource if one-hop replication is not available. One-hop replication is an interesting feature in a communication network, since it contributes to reducing search lengths at a limited cost. It has been sometimes included as part of search mechanisms in P2P networks [16, 18, 61, 78].

The performance of the proposed algorithms has been evaluated using the analytical models, and validating the model's predictions by means of simulation experiments, finding good agreement between them in all cases. Three types of networks have been considered in the simulations (see Section 3.3.4): regular networks, Erdős-Rényi networks, and scale-free networks. We contribute some useful conclusions from the comparison of the performance of a SAW and a simple random walk for searching networks. SAW outperforms simple random walks by obtaining shorter searches on the average, especially in scale-free networks. It has been found that the performance gain is large in

networks without one-hop replication, but it turns out to be sensibly smaller for networks with one-hop replication. This is a consequence of the observed fact that, in one-hop replication networks, an average random walk covers nodes (i.e., visited nodes and neighbors of them) almost as fast as an average SAW. This means that *the effect on average search length of one-hop replication is nearly equivalent to that of trying not to revisit nodes*. This observation suggests that SAWs are an interesting alternative to simple random walks for searching networks without one-hop replication, which avoids the overhead of updating information about the neighbors' resources.

1.4.2. Partial Walks

This dissertation proposes an application to resource location of the technique of concatenating partial walks (PW) available at each node to build random walks. A PW is a precomputed random walk of fixed length. Two variations are considered, depending on whether the search mechanism first randomly chooses one of the PWs in the current node and then checks its associated information for the desired resource, or it first checks all PWs in the node and then randomly chooses among those with a positive result. Both of these variations may use PWs that are simple random walks (RW) or self-avoiding random-walks (SAW), resulting in four mechanisms referred to as *choose-first* PW-RW or PW-SAW, and *check-first* PW-RW or PW-SAW, respectively. Our mechanisms assume the use of Bloom filters [12, 14] to efficiently store the set of resources (not their owners) held by the nodes in each partial walk. The compactness of Bloom filters comes at the price of possible *false positives* when checking if a given resource is in the partial walk. False positives occur with a probability p, which is taken into account in our analysis. These assumptions provide generality to our model, since a probability of $p = 0$ models the case in which the full list of resources found are stored (instead of using a Bloom filter).

We provide an analytical model for the choose-first PW-RW technique, with expressions for the *expected search length*, the *optimal length of the partial walks*, and for the *optimal expected search length*. We found that, when the probability of false positives in Bloom filters is small, the optimal expected search length is proportional to the square root of the expected search length achieved by simple random walks, in agreement with the results in [24]. Another interesting finding is that the optimal length of the partial walks does not depend on the probability of false positives of the Bloom filters. We also provide analytical models for the choose-first PW-SAW mechanism as well as for the check-first variations, which predict their expected search length. Then, the predictions of the models are validated

by simulation experiments in the three types of randomly built networks as described in Section 1.4.1. These experiments are also used to compare the performance of the four mechanisms, and to investigate the influence of parameters as the false positive probability and the number of partial walks per node. Finally, we have compared the performance of the four search mechanisms with respect to simple random walk searches. For choose-first PW-RW we have found a reduction in the average search length ranging from around 98% to 88%. For choose-first PW-SAW such a reduction is even bigger, ranging from 12% to 5% with respect to PW-RW. Check-first PW-RW and PW-SAW can achieve still larger reductions increasing the number of PWs available at each node.

1.4.3. Partial Walks with Dynamic Resources

In this work, a version of the PW-RW mechanisms described in the previous section with deterministic (full) information storage is applied to networks where resources are dynamic. In particular, we consider a scenario in which several instances of each resource are randomly placed in the nodes across the network. Each instance of a given resource may appear and disappear over time. All the nodes of the network may launch independent searches for different resources (e.g., files), and we are interested in measuring the average performance of searches between *any* pair of nodes.

In this context, we have developed an analytical model both for the choose-first PW-RW and the check-first PW-RW searching mechanisms. Expressions are given for the corresponding *expected search length* of each mechanism, as a function of several parameters of the model such as the network structure, the resource dynamics, and the setup of the searching mechanism. Then, as before, the predictions of the models are validated by simulation experiments in three types of randomly built networks: regular, Erdős-Rényi, and scale-free. These experiments are also used to compare the performance of both mechanisms, and to investigate the influence of the resource dynamics. We found that both search mechanisms provide optimal search lengths that are small, and they do not depend heavily on the resource dynamics or on the network type. Finally, we have also compared the performance of the proposed search mechanisms with respect to random walk searches. For the choose-first PW-RW mechanism we have found a reduction in the average search length with respect to simple random walk ranging from around 57% to 88%. For the check-first PW-RW mechanism such a reduction is even bigger, achieving reductions above 90%.

1.5. Dissertation Organization

After this introductory chapter, the rest of the dissertation is organized as follows. Chapter 2 provides a study of the state of the art of the problem of resource location in networks. Then, Chapter 3 gives the definitions, assumptions and notation that are used in the resource location techniques presented in the dissertation. Chapter 4 presents the techniques based on SAWs. Chapters 5 and 6 present the mechanisms based on PWs applied to static networks and to dynamic networks, respectively. Finally, Chapter 7 presents the conclusions of the dissertation and describes research lines for the future.

Chapter 2

Resource Location in Complex Networks

This chapter presents several current approaches to solve the resource location problem in computer networks. The focus is set on solutions used in several P2P networks, as they are a significant paradigm of systems in which it is not desired to rely on centralized services. The techniques described are thus applicable to any other distributed system in which information on resources is not available, as it is assumed by the mechanisms proposed in this dissertation. First, ideas used in structured systems are discussed. Structured systems enforce a particular network topology which is designed to facilitate the location of resources. While generally achieving very efficient searches, this type of systems incur in the processing and bandwidth overhead derived from setting and maintaining the necessary network topology. Then, approaches used in unstructured systems are explained. Unstructured systems do not impose any restriction on the network topology, thus avoiding the mentioned overhead. The resource location mechanisms presented in this dissertation belong to this type of systems. Then, we examine in more detail other works that use random walks to locate resources in networks, pointing out the differences with the mechanisms proposed in this dissertation.

2.1. Structured Systems

In structured systems, each resource is assigned to a unique node by an algorithm. This node is then responsible for knowing where the resource is located in the network. It typically keeps a pointer to the node that actually holds the resource. When searching for that resource, the same algorithm is executed to find the node which is responsible for it. The search is then

directed to that node, which provides the location of the resource, resulting in short search lengths. This section explains this idea, outlines its benefits and drawbacks, and finally describes some proposals for structured systems that use it.

2.1.1. Distributed Hash Tables

The assignment of the responsibility for resources to nodes is usually performed using a space of identifiers or keys. Each resource has a key, that can be calculated using a hash function (e.g., SHA-1 [30]), taking its name, and/or other attributes as input. On the other hand, network nodes also have an identifier, which can be calculated, for example, applying the hash function to its network address.

Each node is responsible for a partition of the space of resource keys, knowing the location of the resources whose keys belong to that partition. The system performs the mapping between the resource keys and nodes identifiers. When a node asks for a resource, its key is computed and the system returns the node responsible for that key. That node is then queried for the actual location of the resource.

Because of the operation described above, these systems are know as *Distributed Hash Tables* (DHTs). Real systems differ in aspects like the way the space of keys is partitioned, how it is reassigned when nodes come into or go out of the network, and the routing of searches through the network.

DHTs are based on constructing a particular network topology with the participant nodes. It is assumed that the nodes are connected to a *physical* network which provides full connectivity. Then, an *overlay* network is built on top of the physical network (which is also referred to as the *underlying* network). This overlay network is a logical network where each node chooses its neighbors among all network nodes. The topology of the overlay is designed to support efficient searches. Several examples from real structured systems are presented later in this section.

In general, structured systems based on DHTs have the following desirable properties:

- The number of communication steps to find a resource (or *hops* in the terminology of searches based on random walks, introduced in Chapter 3) is lower than in unstructured systems. Typically, resources are found in $O(\log(N))$ where N is the number of network nodes. Therefore, searches impose little communication and processing overhead on the network.

- A resource is always located if it is present in the network and if the system is in a stable state. In other words, there are no *false negatives*, unless the system is subject to a high churn of nodes.

Some proposed structured systems well known in the literature include Chord [84], Pastry [79], Tapestry [91], CAN [71], Kademlia [62] and Viceroy [60]. The remainder of this section describes Chord, Can and Kademlia. FreeNet is also included here as an example of loosely structured systems, a relaxed variation of the structured system paradigm. The section concludes with a review of the disadvantages of structured systems, that can make unstructured systems more appropriate under some conditions.

2.1.2. Chord

Chord [84] is based on a simplified version of the distributed data location protocol by Plaxton et al [70]. In this version of the protocol, concurrent failure and joins of nodes are appropriately handled. Nodes identifiers and resource keys are computed using *consistent hashing* [44]. This technique guarantees that resource keys are distributed evenly among network nodes.

Node identifiers belong to a space of 2^m identifiers, where m is the size of identifiers in bits. These identifiers are ordered in a circle modulo 2^m, and nodes are given one of these identifiers as they enter the network, that is, nodes are placed in the mentioned circle. The space of resource keys has the same size as the identifiers space. Then, a resource key k is assigned to the first node whose identifier is equal or greater than k, that is, the first node found in the circle moving from k in clockwise direction. That node is called the *successor* of k ($successor(k)$).

Figure 2.1 shows an example of the circle of identifiers for $m = 3$. There are three nodes and five keys. Key 2 is assigned to node 2, existing in the circle. On the other hand, key 3 is assigned to node 5, which is the first node found clockwise from position 3, where is no actual node. The same algorithm assigns keys 6 and 7 to node 0, since no nodes 6 and 7 exist in the circle.

Nodes also store information to route searches. This information is kept as the *finger table* of each node, which has a maximum size of m entries called *fingers*. The entry in the i^{th} position of node n is filled with the first node s whose identifier is at least at distance 2^{i-1}, following the circle of identifiers. In other words, $s = successor(n_{id} + 2^{i-1})$, where n_{id} is the identifier of node n. Each entry in the finger table corresponds to a range of the identifiers space: $[(n_{id} + 2^{i-1}) \bmod 2^m, (n_{id} + 2^i) \bmod 2^m)$. The node in that entry is

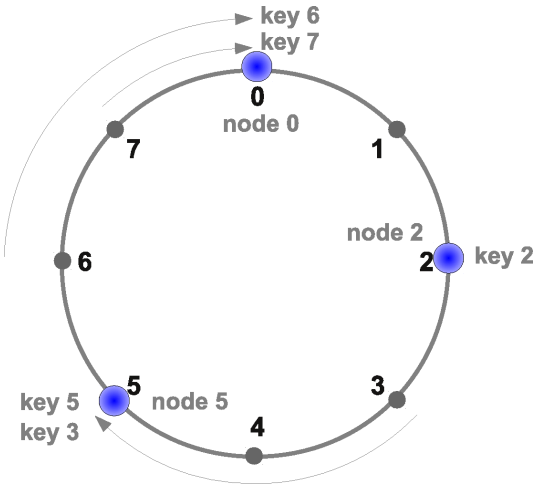


Figure 2.1: Circle of identifiers in Chord for $m = 3$ bits.[1]

the first node in that range in clockwise direction.

Figure 2.2 shows the finger tables of the nodes in the previous example. Focusing on node 5, ranges are $[5 + 2^0 \bmod 2^3, 5 + 2^1 \bmod 2^3)$, $[5 + 2^1 \bmod 2^3, 5 + 2^2 \bmod 2^3)$ and $[5 + 2^2 \bmod 2^3, 5 + 2^3 \bmod 2^3)$, that is, $[6, 7)$, $[7, 1)$ and $[1, 5)$.

When node n wants to locate a resource with key k, it routes the search as follows. If it knows $successor(k)$, it directs the search to that node. Otherwise, it looks in its finger table for node j whose identifier immediately precedes k. It then asks j for the node closest to k. This process is repeated iteratively, with n getting information about a node closer to k in each step. Complexity of this algorithm is shown to be $O(\log N)$, where N is the network size.

Finger tables become stale when nodes leave the network, or when new nodes join. In these cases, tables must be updated and resource keys reassigned to nodes, incurring in a significant overhead.

2.1.3. CAN

CAN [71] (Content-Addressable Network) is a DHT system that models the identifier space as a d-dimensional cartesian space. The space is divided into zones (e.g., rectangles if $d = 2$), and each zone is assigned to one node. Zones do not overlap and cover the whole identifier space. Figure 2.3 shows an example of a 2-dimensional identifier space split into three zones.

[1]The figures in this chapter are adapted with permission from [76].

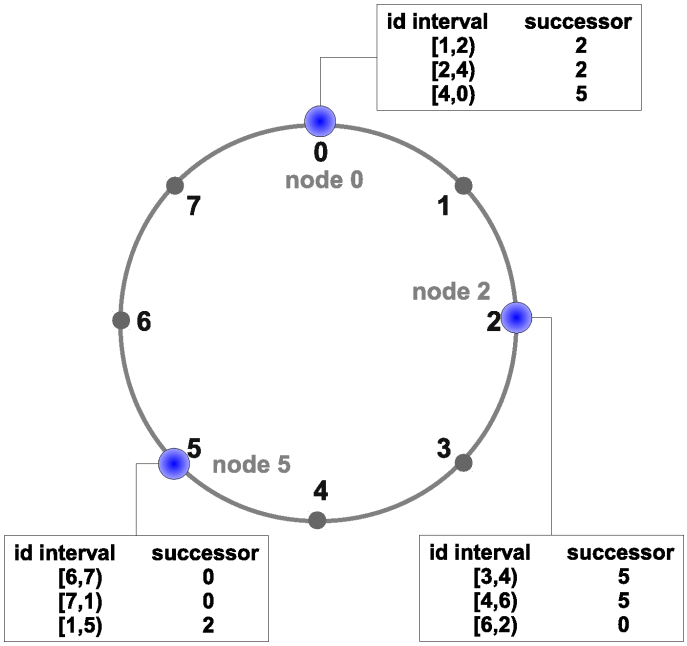


Figure 2.2: Finger tables in Chord for $m = 3$ bits.

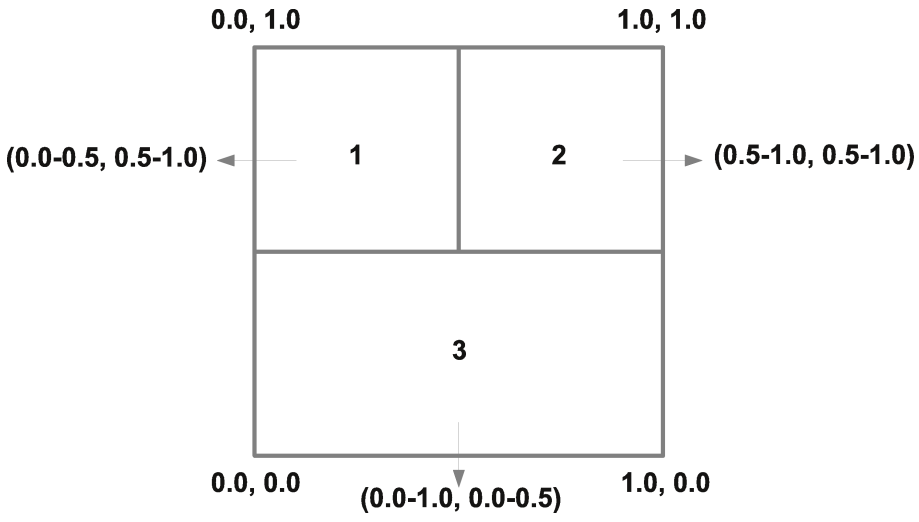


Figure 2.3: A 2-dimensional CAN identifier space.

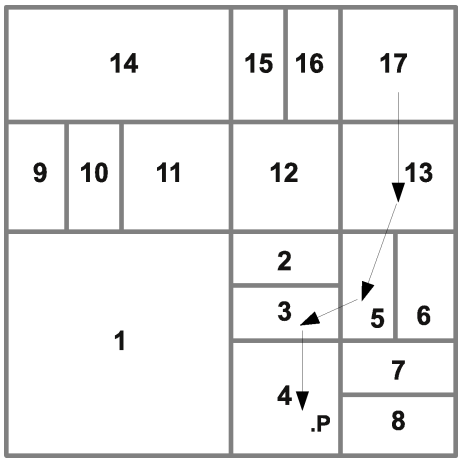


Figure 2.4: A search in a CAN identifier space.

A key-value pair (K, V) is assigned to a node as follows. The key K is mapped to a point P in the cartesian space by means of its coordinates using a uniform hash function. The pair (K, V) is stored in the node which was assigned the zone where the point P lies. When searching for key K, the hash function is used again to obtain point P and a search message is sent to the node in charge of the zone containing P.

Routing is made possible because each node keeps a list of its neighbors, i.e., the nodes responsible for the zones adjacent to its own. Messages carry a destination address in the form of P, a point in the coordinate space. Then, a node receiving a message forwards it to the neighbor whose zone is closer to P_d. The number of hops to find the resource is $O(n^{1/d})$. Figure 2.4 illustrates the routing of a search through the coordinates space, until it reaches the node that owns the zone where the point P lies.

Departure and recovery of nodes imply a reorganization of the zones, which is performed by merging and partitioning zones, respectively.

2.1.4. Kademlia

Kademlia [62] is one of the most recent structured systems and it is gaining acceptance. In fact, implementations of the algorithm can be found in Kad (developed originally by the eMule community to replace the server-based version of the eDonkey [27] network), Overnet [68] and BitTorrent [11], among others.

Kademlia is a DHT that uses a XOR metric to calculate the distance between two identifiers: $d(x, y) = x \oplus y$. The nodes of the network are

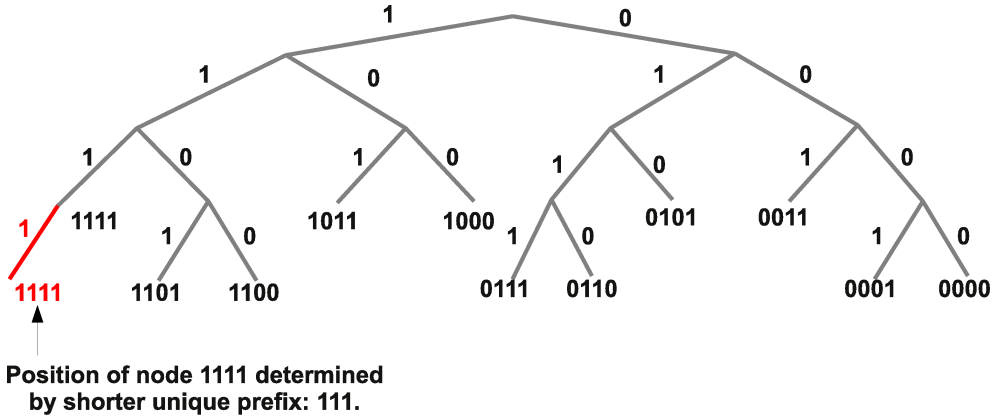


Figure 2.5: Nodes as leaves in a Kademlia binary tree.

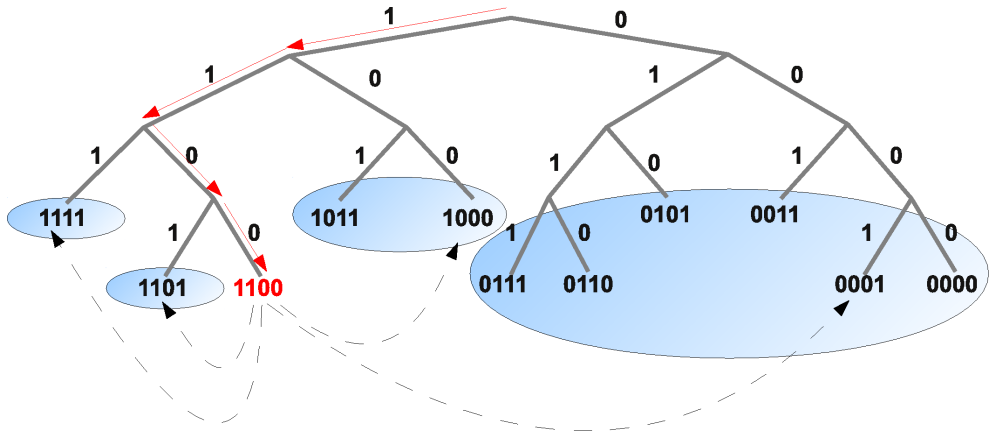


Figure 2.6: Successive Kademlia subtrees for node 1100.

organized as the leaves of a binary tree. The position of a given node is determined by the shortest unique prefix of its identifier. Figure 2.5 shows an example where node identifiers are four bit long.

Suppose that we follow the path from the root of the tree to a given node. At each step, the tree is divided in two. Each node is associated with several subtrees in the network. These subtrees are determined by choosing the subtree that does not contain the node at every step from the root to the node. This is illustrated by Figure 2.6 for node with identifier 1100.

Let l be the size of identifiers in bits. Then, the distance between the considered node and any node each subtree is between 2^i and $2^{i+1} - 1$, with $0 \leq i < l$. Also, for any given subtree, the distance between two nodes

belonging to it is always shorter than the distance between a node of the subtree and a node outside the subtree.

The systems guarantees that each node knows at least one node in each of its subtrees. Going back to Figure 2.6, dashed lines show an example of nodes that could be known by node 1100: 1111, 1101, 1000 and 0001.

The responsibility for a resource is assigned to the node whose identifier is closer to that of the resource. When a node desires to find a resource, it first computes its identifier and then it uses the XOR metric to find out which of its known nodes (one for each of its subtrees) is closer to the resource. Then, it sends a query message to that node. When the node receives the query, it will compute the XOR metric between the resource and each of its known nodes, replying to the original node with the identifier of the closest node. Then the original node sends a query to that node and the process is repeated iteratively until the node that knows about the resource is located. Since it is a binary tree, each iteration reduces the distance at least by 1/2.

Consider, for example, that node 1100 wants to find a resource with identifier 1010. It applies the XOR metric to all the nodes it knows about:

- $1010 \oplus 1111 = 0101 = 5$
- $1010 \oplus 1101 = 0111 = 7$
- $1010 \oplus 1000 = 0010 = 2$
- $1010 \oplus 0001 = 1011 = 11$

It now sends a query message to node 1000, since the XOR metric is lowest for it. Node 1000 then replies with the identifier of a node that it knows it is closer to the resource: 1011. Finally, the node 1100 sends its query to node 1011, which is the one that knows about the resource (the closest to it in the entire system), and sends back a message with the resource location. This process is illustrated in Figure 2.7.

The actual operation of Kademlia is a little more complex. Nodes keep a list of nodes (instead of a single one) for every associated subtree. They manage insertions and deletions from that list depending on since how long the node has received a message from each node in the list. Older nodes are given precedence, since it is more likely that will continue up in the future [81]. The mechanism by which Kademlia handles departures and entries of nodes is related with this. Details can be found in [62].

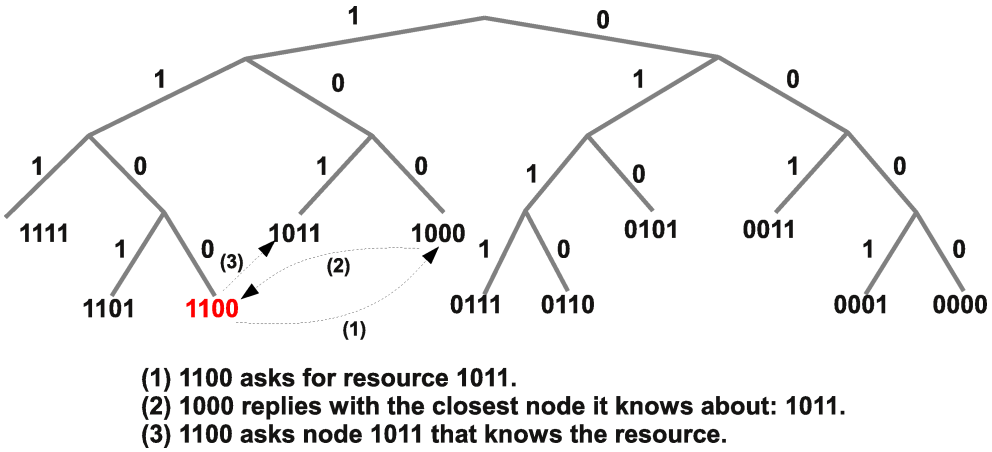


Figure 2.7: Node 1100 searches for resource 1010.

2.1.5. Loosely Structured Systems

Loosely structured systems are not true structured systems, while they use some of their ideas. The general approach is to route searches by means of heuristics that use local information. However, it is not guaranteed that the resource will be found using those heuristics.

FreeNet [19, 20] is an example of loosely structured systems. It is focused on privacy and anonymity of its users. The objective is to make the system censorship-proof, while maintaining a good performance in efficiency, availability, etc. FreeNet uses unique identifiers for nodes and for resources. Besides, nodes keep a table of their neighbors and their identifiers. When a search gets to a node, it compares the resource identifier with those of its neighbors, forwarding the message to the neighbor with closest identifier. While this approach is similar to those of the structured systems described so far, the difference lies in that, in FreeNet there is no strict control over the topology of the network. In true structured systems, the topology of the overlay network (a ring, a tree, etc.) is forced to be such that it exhibits some properties that ensure that resources are always located. Such restriction is not present in FreeNet, and that is why it is considered a loosely structured system.

2.1.6. Disadvantages of Structured Systems

Due to the efficient searches achieved by structured systems, they have been applied in some P2P systems, for example in some file storage solutions like OceanStore [47], Scan [17] (which is based on Tapestry) or Past [26]

(based on Pastry). However, DHTs have not been widely accepted in file sharing systems (with the exception of Kademlia and its recent incorporation to eDonkey clients) because of the drawbacks they present. A description of these drawbacks can be found in [6], [16] and [36] and are extracted here:

- **DHT tables maintenance is costly.** Even if the execution of the algorithm itself is not very expensive, the scheme is broken when nodes abandon the system or enter into it. This forces a re-execution of the algorithm, incurring in communication overheads, the more relevant the higher the rate of churn of the nodes.

- **Potential unfairness of resource assignment.** The resource placement algorithm does not in principle take into account the popularity of the resources being assigned and the computing power and load of the nodes. Resource popularity has been observed to vary largely in content-sharing P2P networks [35]. This may lead to low processing power nodes being assigned popular resources, which means high load due to searches. Load balancing mechanisms can be added to take these factors into consideration [34], at the cost of extra computing and communication overheads.

- **Resource replication due to popularity is not exploited.** If a popular resource is downloaded and cached by many nodes, the number of instances of that resource increases, making it easier to find the resource in future searches. However, the resource placement mechanisms always directs searches to the only node that has the responsibility for that resource. A mechanism to cache queries has been proposed to increase the probability that popular resources are cached. Kademlia uses this type of mechanism.

- **Queries by keyword are not supported.** When searching for a resource, the algorithm needs the same input it used to assign that resource to a node, typically the name or identifier of the resource. However, nodes may not know the exact name of the resource, or may want to do keyword searches, and this is not naturally supported. Structured mechanisms can be supplemented with keyword indexes to overcome this problem [72], again incurring in additional computing and communication costs.

- **Locality is not preserved.** A structured system ignores the origin of a resource when it is assigned to a network node. Thus, resources from the same node will typically not be co-located in the network, losing

the advantages that locality may provide when searching and browsing for resources. SkipNet [36] proposes a solution to preserve locality.

- **Application level information is not considered.** Relevant information can be embedded in the name of a resource, for example, the position in a hierarchical structure. This information is also ignored when the resource placement algorithm assigns the responsibility of that resource to a node.

2.2. Unstructured Systems

Unstructured systems are characterized by (1) not having a control mechanism over the placement of resources, and (2) not imposing any restriction on the way nodes connect to each other (the overlay network). Generally speaking, these characteristics yield the following desirable properties:

- They have little communications overhead to manage the arrivals and departures of nodes.
- They take advantage of the natural replication of popular resources.
- They allow to make queries by keyword instead of by exact matching in a much simpler way than in structured systems.

Unstructured systems can be *centralized* or *decentralized*. Centralized systems are based on a central server that keeps an index of the resources held in all nodes. Decentralized systems do not rely on a central server, using techniques based on *flooding*, *supernodes* and *random walks*. This section describes centralized systems, systems based on flooding and systems based on supernodes. Searches based on random walks are covered separately in Section 2.3. Then, Section 2.4 describes proposals involving self-avoiding random walks. Finally, Section 2.5 covers proposals using the concept of partial walk. An interesting comparison among search mechanisms in unstructured systems is provided in [87].

2.2.1. Centralized Systems

This is the simplest type of unstructured systems, but also the least interesting if we pursue having a network with no need of central infrastructure that needs to be configured and maintained. At some initial time, each node registers with the central server and notifies it about the resources it holds (and that it wants to offer to the network). When a node wants to find a

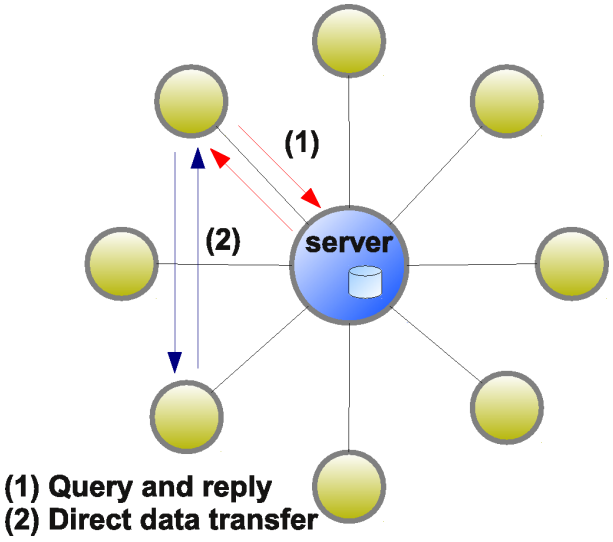


Figure 2.8: A search in a centralized system.

resource, it queries the central server, who replies with a list of nodes holding an instance of the desired resource. Then, the node communicates with the chosen holder of the resource to do the data transfer. This a typical client-server relation combined with direct interaction between nodes, in what is sometimes called *hybrid systems* [6]. Figure 2.8 illustrates this behavior.

Centralized systems are easy to implement, and searches are fast and efficient. But they have a single point of failure, which makes them vulnerable to failures, attacks, censorship, etc. Furthermore, centralized systems are inherently non-scalable, since they are limited by the capacity of the server. Napster [65] is an example of centralized P2P system.

2.2.2. Resource Location Based on Flooding

Flooding is a simple search mechanism that was originally used in contents-sharing systems like Gnutella [33]. When a node receives a search message, it checks if it has the resource looked for. If so, it replies to the node origin of the query. Otherwise, it forwards the message to all its neighbors except the one through which the query arrived. This mechanics is then a *breadth-first search* (BFS).

The scope of the search can be easily limited by a Time To Live (TTL) mechanism. The TTL value is decremented in each step before the message is forwarded. When the TTL reaches zero, the message is discarded. The election of the TTL is relevant. A value too low would lead to unsuccessful

searches. A value too high would easily collapse the network because of the bandwidth demanded by search messages. In general, it is accepted that flooding mechanisms do not scale well [41, 75]. It is not difficult to show that the number of messages grows exponentially with the value of the TTL parameter.

Improved Flooding

Several refinements have been proposed to the basic BFS mechanism to improve its performance [89]:

- *Do not re-forward searches already forwarded.* A simple way to decrease the number of messages consists of having each node keep a list of recent searches forwarded, so that it will not forward again a message of a search that has already been forwarded.
- *Iterative Deepening or Expanding Ring.* In this mechanism, several flooding waves are launched sequentially with increasing values of the TTL, until the resource is found or the maximum TTL is reached.
- *Directed BFS.* The source node sends the search message only to those nodes which it believes will have reasonable chances of finding the resources. Their neighbors will then forward search messages using normal flooding.
- *Local indexes.* In this technique, nodes keep a list of neighbors within a given number of hops and the resources they held. Searches are processed only at specific depths specified by a policy.

Another work to improve flooding mechanisms can be found in [43]. They propose two variations of BFS:

- *Forward search messages only to a subset of neighbors.*
- *Intelligent search mechanisms based on past results.* Nodes keep a profile of each of its neighbors, based on the successful searches achieved by them. This is used to rank the neighbors. When a search is received, it is forwarding to the highest neighbors in the ranking.

The *Distributed Resource Location Protocol* [63] (DRLP) combines the following techniques: (1) when a search message arrives, it is forwarded to each of its neighbors with a probability p, and (2) when a search finds the resource, a message is sent back to the source along the same path it traversed. Each node in the path stores the location of the resource in a cache

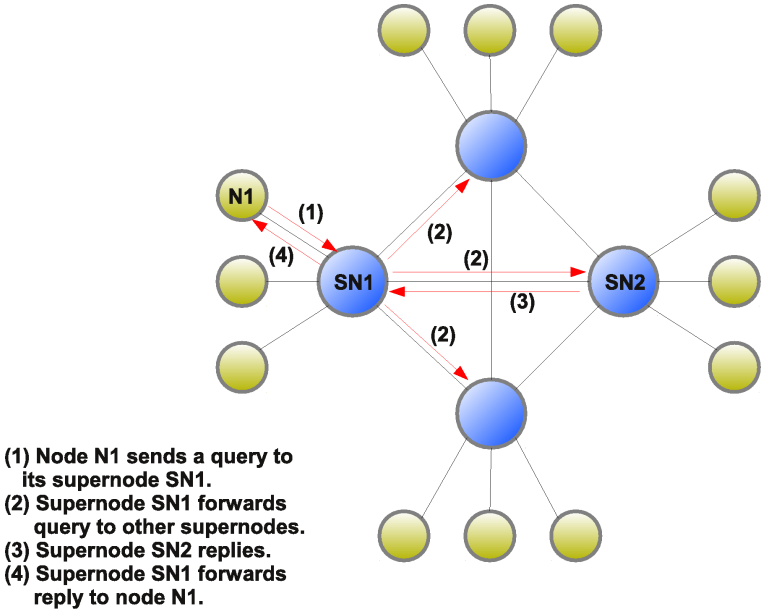


Figure 2.9: A search in a supernodes system.

for future searches. While DRLP decreases the number of search message with respect to normal flooding, two problems remain open: the computation of an appropriate p (since resource replication is not known), and the limited size of caches.

The solutions described above do improve the performance of flooding. However, they do not solve the scalability problem, since they all rely on BFS as their basic mechanism (they still grow exponentially with the TTL value). This is the reason why other solutions have been proposed, like those based on supernodes, which are covered in the next section.

2.2.3. Resource Location Based on Supernodes

The main idea behind supernodes is to let P2P systems benefit from the heterogeneity of their nodes allowing the introduction of two roles: the normal nodes, which provide resources and demand them, and the supernodes, which keep indexes to resources. Each normal node is connected to at least one supernode. In addition, supernodes are connected among them. This latter subnetwork can be regarded as a true P2P system. Figure 2.9 illustrates a system based on supernodes.

However, this duality of roles moves away from the idea of a pure P2P system towards the notion of a centralized system. However, the approach

stands somewhere in the middle, exhibiting some interesting features:

- Searches are much faster than other solutions for unstructured systems (e.g., flooding), still avoiding to have a single point of failure.
- The system takes into account and benefits from the heterogeneity of nodes: most powerful ones will became supernodes and perform more tasks to serve searches in the entire network. In pure P2P systems all nodes are considered equal in terms of sharing tasks and loads, regardless of their differing capacities.

On the other hand, supernodes systems have also weak points. For example, they are vulnerable to supernodes failures (technical problems, attacks, censorship, etc.). When this happens, other nodes need to take up their responsibilities for the system to keep functioning. However, not any node is appropriate to become a supernodes. It needs to have enough capacity and bandwidth, and the willing of its owner to assume the role of supernode. For example, if not enough capacity is available, the new supernode will become a bottleneck for searches. Even taking this in consideration, the damage to the system is not as high as it would be in a centralized system, since supernodes, by definition, are not a single point of failure.

One of the supernodes system proposed in the literature is *Cluster-based Architecture for P2P* [46] (CAP). In CAP, the network is divided into clusters of nodes. For every cluster, a *delegate* is chosen. Queries are sent to the delegate within the cluster. Only when it does not know about the resource, the query is sent to other delegates outside the cluster. While the basic notion of supernode is present in this idea, CAP has important limitations due to centralization. There is a unique *clustering server* that assigns a cluster to a joining node, notifying it the delegate of the cluster. In addition, node capacity considerations are not taken into account when choosing delegates.

In [9], a system is proposed that uses flooding to forward queries among supernodes. On the other hand, supernodes in the eDonkey [49] network do not interact at all. Another real system based on supernodes is Kazaa [45], although their technical details have not been published.

2.3. Resource Location Using Random Walks

As mentioned in Chapter 1, the resource location mechanisms proposed in this dissertation are based in random walks (RW). The basic idea is as follows. When a node wants to locate a resource, a search message is sent to one of its neighbors chosen uniformly at random. That node checks for the

resource and if it does not know about it, another search message is sent to one of its neighbors. This process is repeated until a node that knows about the desired resource receives a search message. Therefore, we can say that resource location uses messages that follow a RW through the network until the resource is found, or the search message is discarded by some TTL (Time To Live) mechanism. Such a search message proceeding through the network is known as a *walker*. In a resource location mechanism, the starting node can launch a single walker or several k simultaneous walkers. In this latter case, it is referred to as a *k-random walk*.

There are two concerns that immediately arise when considering random walks to perform a search: it may take a large number of hops (and messages) to find the resource, and it may not find the resource at all even if it exists in the network, returning a false negative when the TTL is reached.

An analysis of the performance of RWs is done in [57]. The authors compare the performance of k-random walks with flooding, taking into account factors like the *resources popularity* (the probability that a search is started for each resource), and the *resources replication* (the probability that a randomly chosen node knows about the resource). The authors confirm that flooding incurs in considerable overhead (even if diminished by mechanisms like the expanding ring), that compromise its scalability. They also show that a k-random walk outperforms flooding by up to two orders of magnitude in the number of message needed (using $k = 32$). However, the number of hops needed is larger for random walks. But the authors guess that their effective times are shorter, since the load of the nodes is kept lower than with flooding.

Several works attempt to improve the performance of RWs, like the mechanisms proposed in this dissertation, providing the walker with extra information to direct the search. This section briefly describe some of them.

2.3.1. Guided Searches

Some techniques require nodes to keep information about the past history of searches to be able to guide future searches:

Record of previous searches: Nodes keep information of the searches that they have processed (their identifiers and the neighbor they used to forward the search message). When a walker visits a node that has already been visited by the same search, that node tries to choose a neighbor that has not been used in the past, increasing the probability of covering new regions of the network and hence the probability to

find the resource. This technique was proposed in [57] and reported to achieve up to 30% reduction in number of messages and hops.

Adaptive Probabilistic Search (APS): This technique is a variation of the k-random walk proposed by [86] in which the forwarding decision is *probabilistic* instead of uniformly random. Each neighbor of a node is associated with an index that is updated according to the final result of the searches that were forwarded to that neighbor when looking for a particular resource. Future walkers looking for the same resource will be forwarded using this index information. Simulation results show that this technique achieves higher success rates than simple RWs.

A similar technique is proposed in [22]. In this work, resources can be associated to *topics*, and nodes maintain routing indexes for each neighbor, with information on the number of resources of each topic that can be reached through it.

Equation Based Adaptive Search (EBAS): This technique [10] uses a mathematical model that estimates the necessary number of walkers k and their TTL to guarantee a target performance in terms of bounds on the success rate, delay and message overhead, given the resource replication. When a search is launched, this mathematical model is used to characterize the random walks that will be used, improving the performance with respect to RWs. However, the overhead necessary for the nodes to estimate the replication of the resource can be important in terms of storage and processing capacity in the nodes.

Attenuated Bloom filters: In this work [73], random walks are forwarded using information on the resources in nodes at several distances. This information is stored in a variation of Bloom filters called *attenuated Bloom filters* (ABP). The objective of ABFs, which are arrays of Bloom filters [12, 14], is to diminish the latency to find nearby instances of the resource, forwarding the search message to the neighbor with higher potential to find the resource at a given distance.

2.3.2. Stateless Searches

Other proposed ideas to improve the performance of random walks do not require to keep the state of the searches at the nodes.

Adaptive termination: This simple mechanism proposed in [57] is an alternative to a fixed TTL. It consists of having the walker contact the

source node every certain number of hops to ask if it should continue or stop.

Forwarding to the neighbor with highest degree: This technique [2] assumes that the network has two-hops resource replication, i.e., that every node knows about the neighbors of its neighbors. Then, when searching for a node (results can be easily translated to resources), it is enough that the walker visits a neighbor of the target node's neighbors. A search forwards the walker to the neighbor of highest degree, because it knows many neighbors and therefore the probability to find the target node is maximized.

Resource replication: When there are several instances of a given resource in several nodes of the network, we say that the resource is *replicated*.[2] The higher replication of the resource, the less hops walkers need to find it. The work in [21] studies the impact of resource replication in unstructured systems, providing replication strategies that calculate the optimal number of resource instances based on the popularity of that resource.

2.3.3. Other Approaches

Dynamic adaptive topologies: With this approach, resource location is not improved by acting on the forwarding mechanism but on the topology of the overlay network that supports the searches. The goal is to use the topology that minimizes the length of searches. These techniques try to take advantage of the heterogeneity of the nodes in the network, giving higher degree to nodes with more processing and storage capacity, bandwidth, etc. The adaptation of the topology is independent from the forwarding mechanism, so this idea can be combined with the mechanisms described above. DANTE [77] and Gia [16] are P2P systems that use topologies dynamic adaptation.

Publication and subscription systems: More sophisticated approaches aim to improve resource location with a more proactive approach. On one hand, resources are advertised by their owners, and that information is registered in some nodes distributed in the network. On the

[2] In this context, an instance can be a copy of the resource or just a reference to it (indicating the node that holds the real resource). In this dissertation, we use the term *replication* when a node knows about a resource held by another node, and the term *multiplicity* when there are several (real) instances of the same resource in the network.

other hand, nodes can subscribe to resource categories they are interested in, and that information is also registered in the network. When a node receives information about an advertised resource and has matching subscriptions from other nodes, it notifies those interested nodes. In this type of systems, random walks can play a role in the advertisement and/or the notification mechanisms. A recent example of this approach can be found in [88].

2.4. Resource Location Using Self-Avoiding Random Walks

Random walks on graphs have been extensively analyzed in mathematics [39, 55, 66], where they are typically modelled as Markov chains, leading to many interesting results that include bounds on the *cover time* [4, 8, 29, 42, 64, 92] (i.e., the number of steps to visit all, or a fraction of, the nodes in the graph), and bounds on the *access time* [13, 69] (the expected number of steps to reach node j starting from node i). These results are frequently based on the spectral properties of the adjacency matrix of the graph and of the transition matrix of the random walk. Random walks have also received much attention from the Physics community, since they reflect the dynamics of many natural systems including protein interactions, polymer chains, communication networks, and social networks.

The basic behavior of a random walk can be modified in a number of ways to optimize its performance metrics on networks. Costa and Travieso [23] study the network coverage of three types of random walks: traditional, preferential to untracked links, and preferential to unvisited nodes. They find that the latter is the best strategy in covering the network nodes, in both random (Erdős–Rényi) networks and scale-free networks. Yang [90] studies the search performance of five random walk variations: no-back (NB), no-triangle-loop (NTL), no-quadrangle-loop (NQL), self-avoiding (SA) and high-degree-preferential self-avoiding (PSA). He finds that all algorithms achieve similar performance in random networks, while the self-avoiding walk outperforms the others in scale-free networks and small-world networks. Both results suggest that a random walk that tries to not revisit nodes is an interesting alternative to pure random walks for network search. Our work focuses on using this type of walk for searching for resources in a network. More concretely, we define a walk that chooses the next node to be visited uniformly at random among the unvisited neighbors of the current node. If no unvisited neighbor exists, the next node is chosen uniformly at random

among all the neighbors of the current node. The walk proceeds until it finds the resource searched for. We will refer to such a walk as a *self-avoiding walk* (SAW) in this dissertation.

Our model of SAW is in fact akin to the *true* SAW defined in [5] at short times, becoming a simple random walk at long times. The true (or *myopic*) SAW is defined as the stochastic process which chooses the next node to be visited among the neighbors of the current node with probability proportional to a negative exponential of the number of times visited. Our definition of SAW differs from this one in that the next node is chosen uniformly at random among the unvisited neighbors or, if none, among all neighbors, regardless of how often they have been visited in the past. This means that our SAW model loses its self-avoiding properties at long times, while the true SAW keeps them at all times.

Although our definition of a SAW coincides with those of a walk *preferential to unvisited nodes* in [23] and of a *self-avoiding* (SA) walk in [90], our work differs from them in that we propose an analytical model to predict network coverage and search performance, whereas their evaluations are only based on simulation results. In addition, [23] evaluates network coverage (not network search).

The analysis of the self-avoiding walk with analytical tools is hard since, unlike the pure random walk, it cannot be modelled as a Markovian stochastic process. Therefore, many questions on SAWs have not yet been answered in an analytical manner. In addition, using analytical results in some real scenarios is impractical when the adjacency matrix of the network is too large or simply unknown. In mathematics, a SAW is defined as a random walk restricted not to intersect with itself. Note that this definition is more restrictive than the one established above in networks, since such a walk never revisits a node, having therefore finite length always. We will refer to this type of walk as *strict SAW*, to avoid confusion with the SAW we are interested in to search complex networks. Available results on strict SAWs are compiled in [59]. This work also includes results on a less restrictive version called the *weakly* self-avoiding walk, in which intersections are not disallowed but *discouraged*. In this type of SAWs, the more intersections in a walk, the less probable it is to occur. Our definition of SAW differs from the weakly SAW in that it always tries to avoid already visited nodes.

Works from the mathematical point of view like the ones referenced and others [48, 59, 83, 59], study SAWs in d-dimensional lattices (Z^d). Results include the behavior of the number of SAWs with n-steps in the lattice and of the *mean-square displacement* (the average distance between the end and the origin of a walk). In complex networks, questions have been addressed mostly through empirical approaches. The previously cited works by Costa and

Travieso [23], and Yang [90] use numerical simulations. Mean access times in lattices with embedded scale-free networks providing long-range shortcuts have been obtained in [15]. In [37], strict SAWs are studied in scale-free networks, obtaining approximate analytical expressions for the mean number of SAWs starting from a generic node, and for the average maximum length of such walks over statistically independent networks. Simulation results support their approximate analytical calculations.

2.5. Resource Location Using Partial Random Walks

As mentioned in Chapter 1, some of the resource location mechanisms proposed in this dissertation are based in the idea of efficiently constructing a random walk connecting together shorter random walks readily available, which are called *partial walks* (PW) in this context. This idea has been proposed as a means to efficiently compute random walks that can be used for any application.

Das Sarma et al. [24] proposed a distributed algorithm to obtain a random walk of a specified length ℓ in a number of rounds. A *round* is a unit of discrete time in which every node is allowed to send a message to one of its neighbors. According to this definition, a simple random walk of length ℓ would then take ℓ rounds to be computed. The number of rounds is proportional to $\sqrt{\ell}$. In the first phase, every node in the network prepares a number of *short (random) walks* departing from itself. The second phase takes place when a random walk of a given length starting from a given source node is requested. One of the short walks of the source node is randomly chosen to be the first part of the requested random walk. Then, the last node of that short walk is processed. One of its short walks is randomly chosen, and it is *connected* to the previous short walk. The process continues until the desired length is reached.

While the same idea is used in some of the algorithms proposed in this dissertation, two key differences need to be underlined. First, the random walk used in [24] has a fixed length, whereas the length of the random walks used by the proposed mechanisms is undetermined, since the random walk that supports the search will be stopped only when the desired resource is found. The second key difference is derived from the different application field. While the goal of [24] is just obtain samples of random walks of a given length, our objective is to locate resources in the network. Therefore, the idea of connecting partial walks needs to be complemented by a means of

storing information on the resources held by the nodes visited by each partial walk.

Hieungmany and Shioda [38] proposed a *random-walk-based* file search for P2P networks. A search is conducted along the concatenation of hop-limited shortest path trees. To find a file, a node first checks its *file list* (i.e., an index of files owned by neighbor nodes). If the requested file is found in the list, the node sends the file request message to the file owner. Otherwise, it randomly selects a leaf node of the hop-limited shortest path tree, and the search follows that path, checking the *file list* of each node in it.

Although the objective in [38] is also resource location, our approach requires nodes only to compute random walks, simpler to compute than their shortest-path-trees. Also, they need more storage space since they keep the pairs *resource-owner*. Keeping only resources, we are able to use Bloom filters. Another important difference is that our searches *jump* over partial walks in which the resource is not located, while their searches traverse the selected tree branch, checking the file list of each node in turn.

Chapter 3

Definitions and Notation

This chapter presents the definitions of the main concepts used throughout this dissertation. The general notation used in the development of the analytical models for the resource location mechanisms is also presented and explained. The particular notation of each of the resource location mechanisms is deferred to the corresponding chapter. First, Sections 3.1 and 3.2 cover the general concepts relative to graphs and random walks, respectively. Then, Section 3.3 presents the network models assumed by the proposed mechanisms, and Section 3.4 explains the aspects of searches common to all of the algorithms. The experimental framework used for the simulation experiments is described in Section 3.5. In addition, Appendix A provides an alphabetical glossary of terms, meant as a quick reference for the main concepts and notation used elsewhere.

3.1. Graphs and Networks

The resource location mechanisms proposed in this dissertation are designed to work on networks. Networks can be described as *graphs*, widely studied in mathematics. Then, the elements of the *graph theory* can be applied to networks, constituting a powerful analytical tool. In this dissertation, the terms *network* and *graph* are used equivalently, except where a distinction needs to be made between the object network and its representation as a graph.

3.1.1. Graphs, Vertices, Edges

A graph is a representation of a set of objects some of which are connected together. This concept is used to represent any kind of communication net-

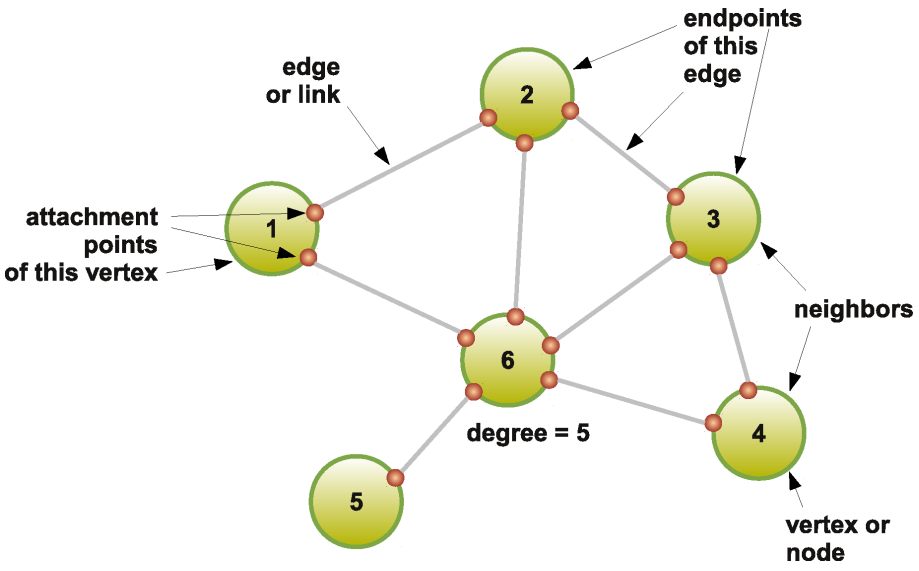


Figure 3.1: An example of a graph.

work consisting of nodes connected by links. In discrete mathematics, a *graph* G is defined as an unordered pair $G(V, E)$ where V is a set of *vertices* or *nodes* and E is a set of *edges*, *arcs*, *lines* or *links* connecting vertices in V. An edge is therefore a 2-element subset of V. Figure 3.1 shows an example of a graph.

Each of the two vertices connected to the ends of an edge or link is called an *endpoint* of that edge. The point of a vertex in which an edge is connected is called an *attachment point* of that verte.. Note that an endpoint belongs to its edge, whereas an attachment point belongs to its vertex. These concepts are also illustrated in Figure 3.1.

3.1.2. Undirected and Simple Graphs. Loops and Multilinks

In particular, the definition above refers to an *undirected* and *simple* graph, which is the type of graph considered in this dissertation [37, 78], since it properly models most communication networks. A graph is undirected when its edges have no orientation. A graph is *simple* when *loops* and *multiple edges* are disallowed. A loop or *auto-link* in this context refers to an edge whose two endpoints are attached to the same vertex. It is not to be confused with a *cycle*, which is any path (sequence of adjacent edges) that return to the same vertex from which it departed. Multiple edges or *multilinks* connect

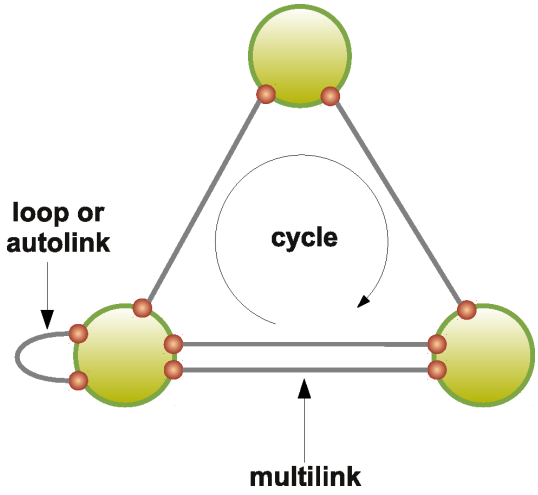


Figure 3.2: Loops, multilinks and cycles in a graph.

the same pair of vertices. Figure 3.2 illustrates the concepts of loop and multilink. The graph shown in Figure 3.1 is undirected and simple.

3.1.3. Finite and Connected Graphs. Network Size

Graphs considered in this dissertation are *finite*, because so is the number of vertices or nodes in a network: $N = |V|$. N is referred to as the *network size*.[1] In addition, graphs in this work are considered to be *connected*, i.e., there exists a path between any two vertices along some edges in the graph. A connected graph describes a network that can be traversed looking for a resource, which is the goal of the mechanisms proposed in this dissertation. The graph in Figure 3.1 is finite (with $N = 6$) and connected.

3.1.4. Degree. Degree Distribution. Average Degree

A given vertex is connected to other vertices by edges. The latter ones are said to be the *neighbors* of the former. In a simple graph, a node has as many neighbors as links are connected to it. This number is called the *degree* of the node and is denote by k. The degree of a node equals the number of attachment points of that node. In the graph in Figure 3.1, the degree of node 1 is 2, the degree of node 2 is 3, etc.

[1] Note that in graph theory, $|V|$ is called the graph's *order*, while $|E|$ is called the graph's *size*. Here *network size* is used for $|V|$, as is usual in the networks community.

The *degree distribution* p_k of a graph is the probability distribution of the degrees over the vertices of the graph. That is, $p_k = n_k/N$, where n_k denotes the number of k-degree nodes in the network (i.e., $\sum_k n_k = N$). Then, p_k is the probability that the degree of a node chosen uniformly at random in the network is k. In our example, we have that $p_1 = 1/6$, $p_2 = 1/3$, $p_3 = 1/3$, $p_4 = 0$, and $p_5 = 1/6$, with $p_k = 0$ for $k > 5$.

The *average degree* of a network is defined as $\overline{k} = \sum_k kp_k$. It is a measure of the *connectivity* of a network, that is, the extent to which nodes are connected among themselves by links. The average degree of the graph in Figure 3.1 is $\overline{k} = 16/6$.

3.1.5. Topology

The topology of a graph or network describes what nodes are connected to other nodes by links. It excludes information on other attributes of links as type, length, bandwidth or cost. In a sense, the topology of a network describes its *overall shape*, excluding other details. Examples of topologies are lattices, rings, trees, stars, etc. These topologies can be found in real networks, and generally correspond to deterministic networks, i.e., networks that are constructed following some deterministic procedure. This dissertation, however, focuses on random networks, so the traditional meaning of topology is not that useful in characterizing a network. When modeling a network, its construction mechanism along with the resulting degree distribution will be taken into account. Network models are presented in a later section.

3.2. Random Walks

The resource location mechanisms proposed in this dissertation are based on random walks in networks. Random walks are extensively studied in mathematics as stochastic processes. However, the application of random walks to resource location, described in this section, introduces some modifications with respect to its mathematical counterpart.

3.2.1. Simple Random Walks

As introduced in Chapter 1, a *random walk* (RW) in a graph is a random algorithm that decides which node to visit next choosing uniformly at random among the neighbors of the current node. The movement of a random walk from a node to the next is called a *hop* in this dissertation. In network

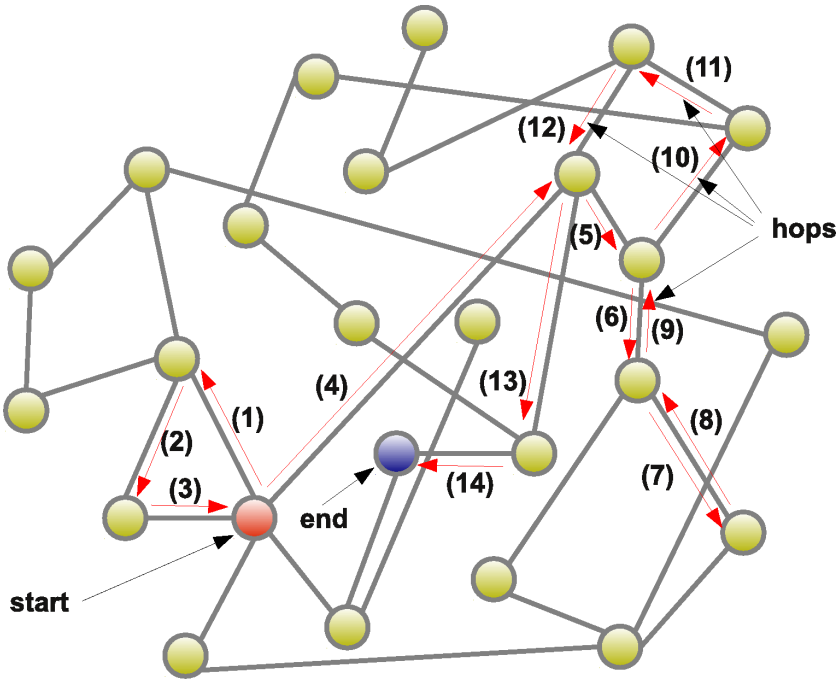


Figure 3.3: A random walk on a network.

terminology, it can be said that a random walk is a particular type of (random) routing algorithm. Figure 3.3 illustrates a random walk on a network. The length of this walk is 14 hops.

The main reason to use random walks to search a network lies in the fact the networks that are considered in this dissertation are in turn random, and the knowledge about the network is desired to be kept as little as possible. As referenced in the Introduction, random walks are analyzed as Markov chains within the theory of stochastic processes.

According to the previous definition, a random walk may visit a node that it has already visited after some number of hops. It can even go back to the node it has just visited prior to the current one. This is shown in the example in Figure 3.3 (hops 1-2-3, and 7-8, respectively). There it can see that the random walk is *blind* or *memoryless* in the sense that the walker does not take into account the past history of the walk when deciding the next node to visit. This, of course, is of little use when the random walk is being used to search the network for a given resource. However, this preserves the randomness of the algorithm, making it a real random walk. In the context of resource location based on random walks, variations of the random walk algorithm can be devised to improve the performance of searches. When

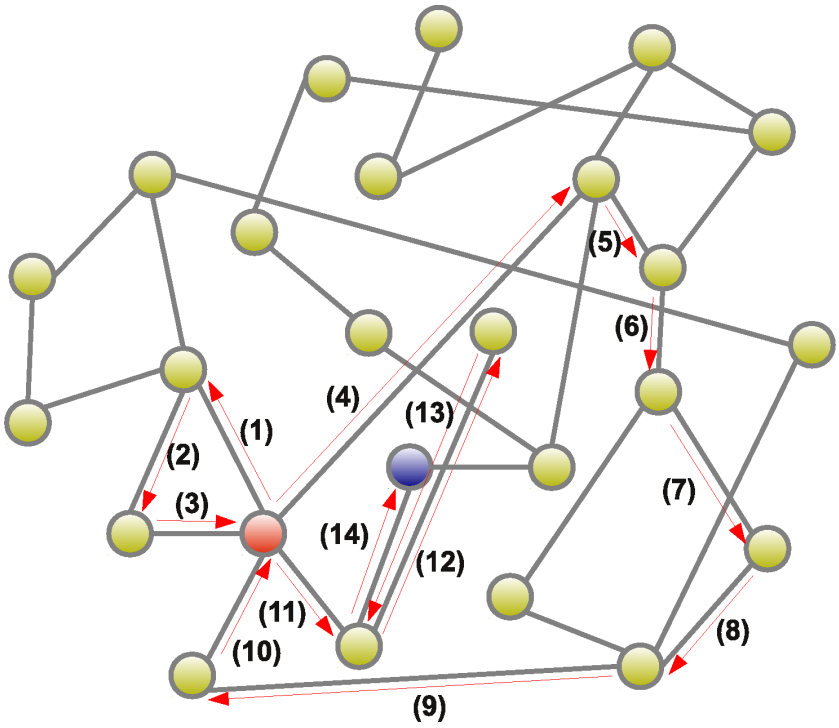


Figure 3.4: A self-avoiding walk on a network.

explicit distinction is to be made between random walks and any of their variations, we refer to the former as a *simple random walk*.

3.2.2. Self-Avoiding Random Walks

In this dissertation, a *self-avoiding random walk* (SAW) in a network is defined as a random algorithm that chooses the next node to visit uniformly at random among the neighbors of the current node that have not been visited yet by the walk. If there are no unvisited neighbors, the walk chooses uniformly at random among all neighbors of the current node, just as a simple random walk would do. Figure 3.4 shows an example of a SAW, based on the RW of Figure 3.3. In this example, all the nodes visited by the walk are new except the following: After hop 2, the walker is in a node of degree 2. Both neighbors have already been visited, so one of them is chosen uniformly at random. In this case, the walker moves forward instead of going backwards. After hop 7, the walker is also in a node of degree 2. However, one of them remains unvisited, so the walker moves forwards, discarding revisiting the node where it comes from (contrary to what the RW of Figure 3.3 did).

After hop 12, the walker is in a node of degree 1, so it has no way out but returning to the node it came from in hop 13. Incidentally, after 14 hops the SAW arrives at the same node as the RW of Figure 3.3 did, just by chance. Even with the mentioned exceptions, this SAW visits 12 new nodes, whereas the RW in Figure 3.3 visited 11 new nodes. This difference becomes larger for longer walks.

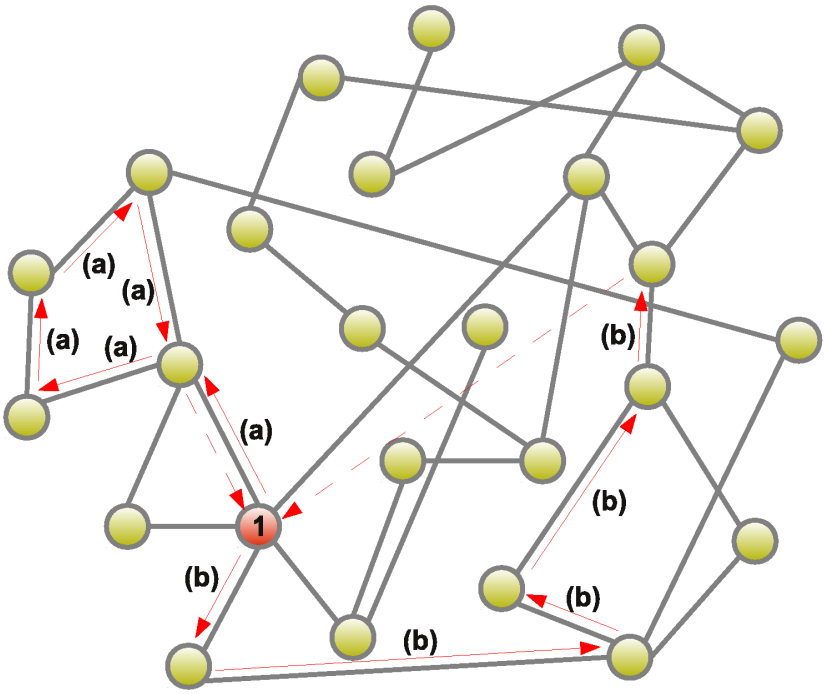
The purpose of this modification to the simple random walk algorithm is clearly to increase the chances of finding the resource by visiting more unknown nodes than a simple random walk. This advantage comes to the price of increasing the information that the walk needs to keep and move along. In this case, this information is the set of network nodes that have been already visited by the walk. This has an impact on the processing power consumed at the nodes and on the network bandwidth used by the search. Both factors, among many others, influence the running time of the search.

3.2.3. Partial Walks

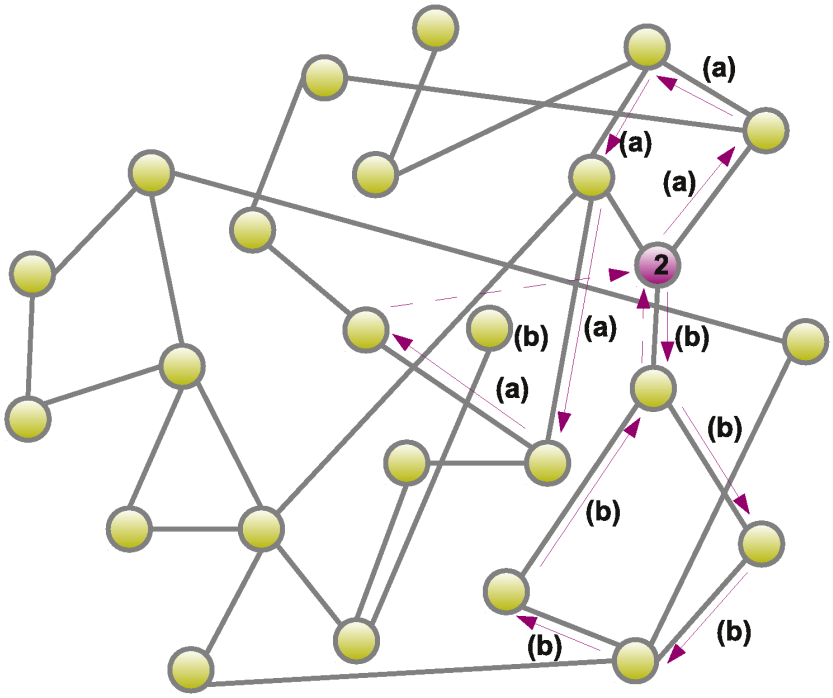
Some of the resource location mechanisms proposed in this dissertation are based on the concept of partial walks. *Partial walks* (PW) are relatively short random walks of a fixed length s that are concatenated to efficiently build a longer random walk. At some initial point in time, every node in the network computes and stores a number w of PWs. That is, each node launches the w random walks, which independently proceed until they reach their defined length s. As a walk progresses, it stores the resources held by the nodes it visits. Then, the walks return to the original node which stores them together with the associated resources. The precomputed PWs of all nodes are then available to be concatenated as necessary by the search mechanisms that will be described in Chapter 5.

Let us consider an example in which every node computes $w = 2$ PWs of length $s = 5$. Figure 3.5(a) shows the two PWs ((a) and (b)) precomputed by node 1. The final dashed arrow for each PW represents the return to the node that is precomputing the PW with the information on the path followed by the walk and on the resources found. Similarly, Figure 3.5(b) shows the PWs of node 2. Note that all network nodes precompute their PWs, although we show just two nodes in this figure.

Suppose now that the node 1 in Figure 3.5(a) wants to locate some resource. It starts a search that chooses one of its PWs, say (b). If the resource is not in that PW (the details of this are mechanism-dependent and are explained in Chapters 5 and 6), another PW is needed to be concatenated to the first PW in its last node. This node happens to be the node 2 of Figure 3.5(b). Node 2 chooses one of its own PWs, say (a), and connects it to



(a) PWs precomputed by node 1.



(b) PWs precomputed by node 2.

Figure 3.5: An example of precomputation of partial walks, for $w = 2$ and $s = 5$.

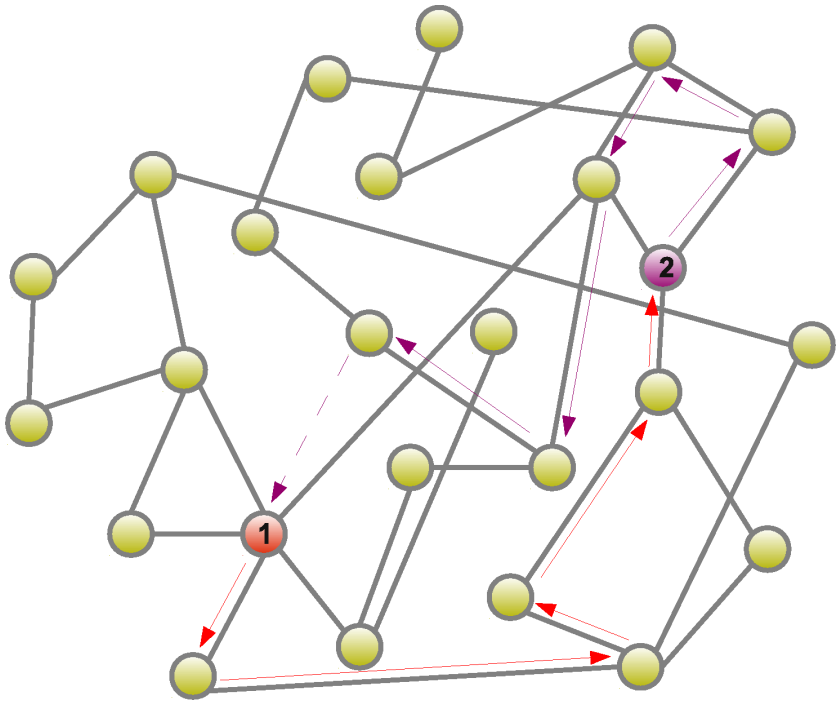


Figure 3.6: A random walk obtained concatenating two precomputed partial walks.

the first PW obtaining a longer random walk that is depicted in Figure 3.6.

The walk used as a PW may be a simple random walk or any of its variations. The algorithms proposed in these dissertations use PWs that are simple random walks or SAWs, which are referred to as *PW-RW* or *PW-SW*, respectively.

The amount of storage needed to record the resources associated with each PW is potentially large. This not only affects the amount of memory consumed at the nodes, which is not a great concern, but it also affects the network bandwidth used as the PW proceeds from node to node. Therefore, it is proposed the optional use of probabilistic data structures instead of the traditional deterministic storage. In particular, the use of Bloom filters is incorporated into the analysis of the proposed mechanisms.

Bloom Filters

A *Bloom filter* [12, 14] is a probabilistic data structure that efficiently stores a set of elements in a fixed storage space. Bloom filters support membership queries. Efficiency in the space usage comes at the cost of introduc-

ing *false positives*. A false positive occurs when the filter is queried about a given element not belonging not stored in the filter (not belonging to the set) and the result is positive. False positives happen with a probability that depends on the number of elements stored in the filter and on the storage space. Therefore, Bloom filters can be designed to achieve a false positive probability considered appropriate for each application.

3.3. Network Models

3.3.1. Overlay Networks

The models for the proposed resource location mechanisms presented in the following chapters assume that searches take place in an *overlay network*. Consider a communication network made up by *network infrastructure elements* (e.g., routers, switches) and *end systems* (e.g., users computers). The Internet and any corporate LAN are examples of this type of networks. The goal of such a network, which will be referred to as the *physical network*, is to provide full connectivity among end systems. This means that a given end system is able to communicate with any other through the network, even if both end systems are not directly connected to each other. To fulfill this mission, we assume that the physical network runs an appropriate routing protocol that provides the routers with optimal routes according to some routing metric (e.g., number of hops, bandwidth, cost, etc.).

Now consider that, for a given application, a subset of the end systems connected to the physical network need to establish logical relations among them, according to some rules. Those relations may be viewed as logical links among those end systems, forming a network *on top* of the physical network. That logical network is called an *overlay network*. It is formed by end systems only (not network infrastructure elements), and its links are just logical relations among those end systems. The *nodes* of the overlay are then the subset of end systems that participate in that application. In this context, the physical network is referred to as the *underlying network*. The neighboring relations among them determine the topology of the overlay network. Of course, neighbors in the overlay are not in general neighbors in the underlying network. Figure 3.7 illustrates the concepts of overlay and underlying networks. Dashed lines represent the links of the overlay, while solid lines represent the physical links of the underlying network.

As described in Chapter 2, many P2P networks (particularly structured systems) use an overlay network, imposing a particular topology that allows to perform efficient searches. Note that, in this context, only end systems may

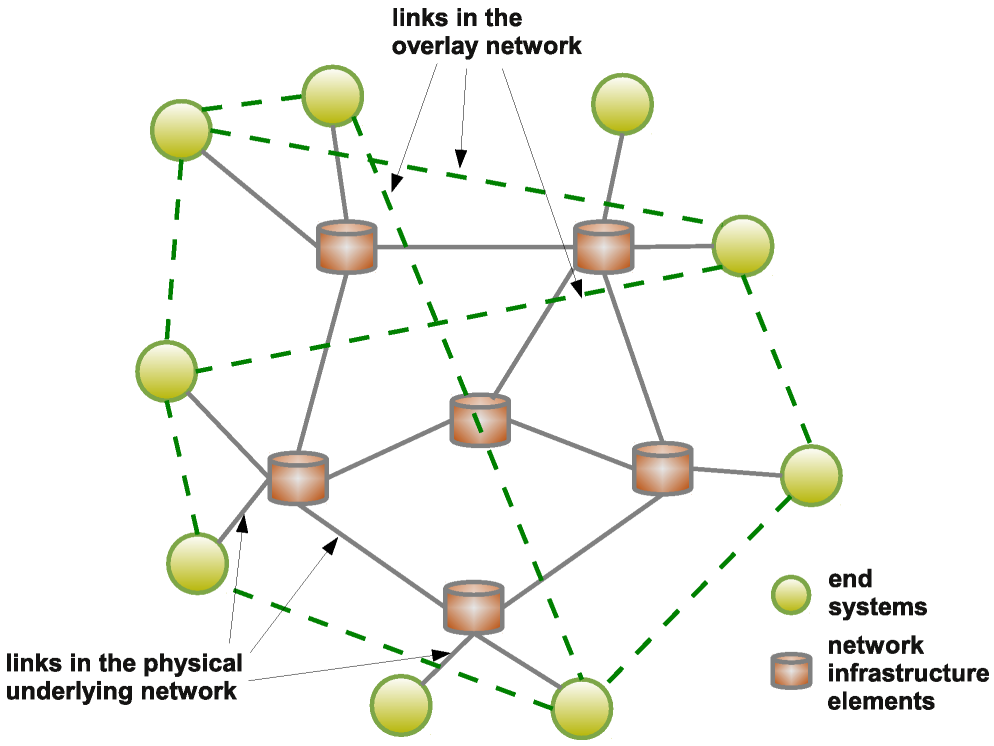


Figure 3.7: An overlay network on a physical underlying network.

hold resources, so it makes sense that only they form the overlay network used for resource location. Our resource location mechanisms, which are targeted for unstructured systems, assume that the network is an overlay network. However, our mechanisms *do not create* the overlay to optimize searches, but just *use any existing overlay* that forms a network whose nodes hold resources and are able to process searches (the end nodes participating in the system).

3.3.2. Random Construction

Graphs or networks are constructed according to a given mechanism that creates nodes and links them to the existing nodes. When the construction mechanism is not deterministic, the resulting graph is called a *random graph*. Note that only the construction mechanism is random: once the network is constructed, it is deterministic in the sense that it is perfectly clear if a link between two given nodes exists or not. According to the characteristics of the graph building mechanism, the resulting random graph will exhibit certain features. In this dissertation we considered random graphs because they

properly represent real networks constructed in a non-deterministic fashion (e.g., unstructured P2P systems, ad-hoc networks).

Because the networks considered can be represented as random graphs, the proposed resource location algorithms are meant to know as little about the network as possible. Therefore, they do not depend on network parameters such as size or topology. However, for the analysis of the algorithms, we assume that the network size (N) and its degree distribution (p_k) are known. This makes possible to assess the performance of the algorithms in several types of networks. It is to be stressed that the algorithms themselves do not depend on this information.

3.3.3. Randomness of Networks

The networks considered in this dissertation have an important additional feature that pertains to their randomness: *each attachment point in each node is linked to another attachment point in the network chosen with uniform probability among the attachment points of the rest of the nodes.*

Consider a set of nodes V with a given desired degree distribution. That is, each node v in V has a known degree k, and consequently has as many attachment points in it. The random construction of the network involves the creation of links. Consider an attachment point a in node v, which we want to link to another attachment point b in the network. Attachment point b is chosen uniformly at random among all the remaining (not already linked) attachment points, with the exclusion of the other attachment points of node v (loops are not allowed), and of the attachment points of all nodes with which v is already linked (multilinks are not allowed). This is, in essence, the mechanism described in [67]. Then, if we create a large number of networks on V, each attachment point will be linked to the rest of attachment points (with the mentioned exceptions) with uniform probability.

The assumption of this feature facilitates the analysis of the proposed resource location mechanisms, which will be presented in Chapters 4 to 6.

3.3.4. Network Types

The proposed resource location mechanisms have been evaluated in three types of networks: regular networks, Erdős-Rényi networks and scale-free networks. Each of these network types is characterized by a particular degree distribution as described below. All networks are assumed to be connected, so that searches can visit all network nodes looking for resources.

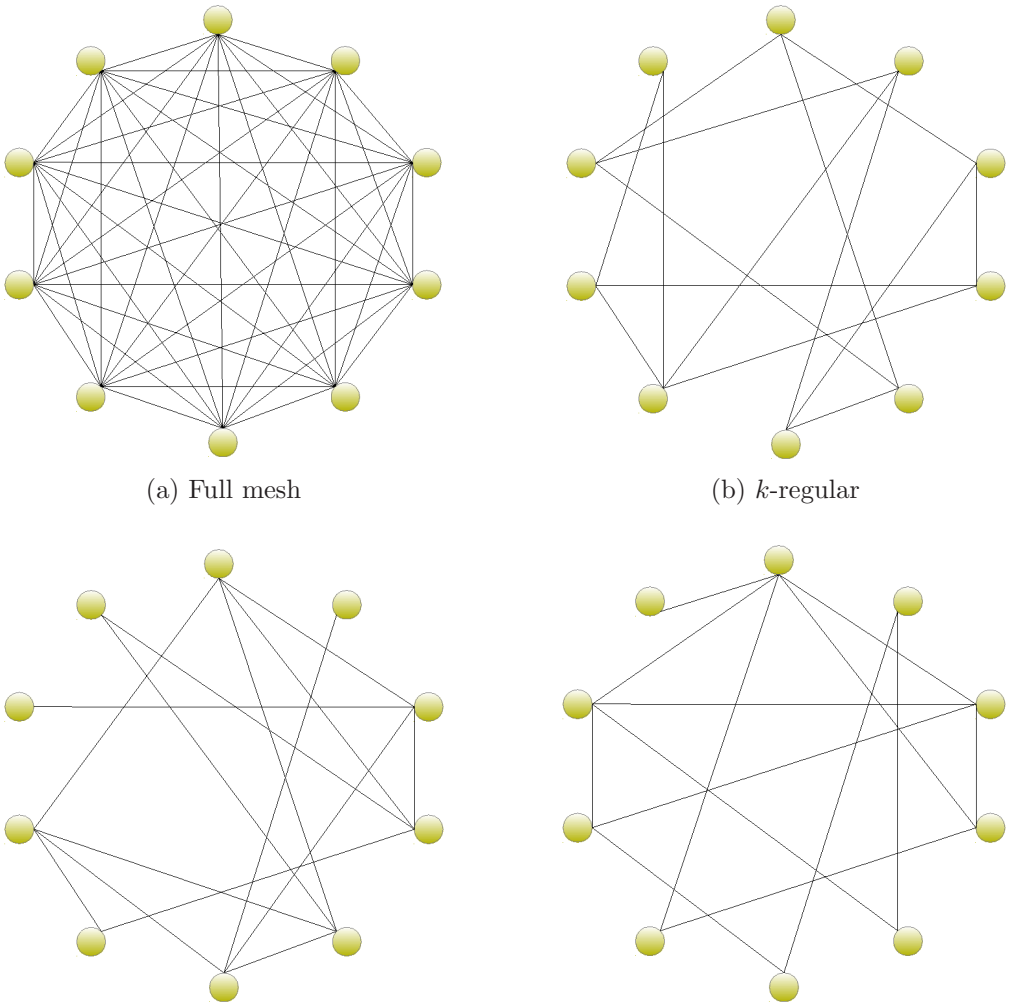
Regular networks: In a regular network, all nodes are connected to a fixed number of neighbors. That is, all nodes have the same degree d. The resulting degree distribution is $p_k = 1$ for $k = d$ and $p_k = 0$ for $k \neq d$. d is referred to as the *degree of the network*. A d-regular network is not to be confused with a *regular lattice*, which may be viewed as a network resulting from replicating a basic cell in one or more dimensions. The topology of d-regular networks is not restricted in any way except in that all nodes need to have exactly d neighbors. In this dissertation, regular networks are used as a baseline case when assessing the performance of the proposed mechanisms.

Erdős-Rényi networks: An Erdős-Rényi (ER) network is a network constructed in the following way. Start from a set of N nodes. For every pair of distinct nodes, a link between those nodes is added with a fixed input probability p (otherwise, the nodes remain unlinked). The resulting degree distribution is binomial, tending to a Poisson distribution for large N when $N \cdot p$ remains constant. In this dissertation, this model has been used as a means to construct random graphs in a natural way. Erdős-Rényi networks are named after Paul Erdős and Alfréd Rényi, who introduced them in 1959 [28].

Scale-free networks: A scale-free network is a network with a degree distribution that follows a power law, at least for large values of the degree k: $p_k \propto k^{-\gamma}$. Many real networks (the Internet, WWW links, P2P networks, social networks, and biological networks) have been observed to have this type of degree distribution [1, 7, 74, 82], with exponents such that $2 < \gamma < 3$. In this dissertation, this model has been used as a way to construct realistic networks to assess the performance of the proposed algorithms.

Figure 3.8 shows examples of the different types of degree distributions just described, for networks of size $N = 10$ and average degree $\overline{k} = 3$. A full mesh is added for comparison. Although the network size is very small, these examples illustrate the relation amont the different degrees. In the k-regular network shown in Figure 3.8(b) all networks have degree $k = 3$. The Erdős-Rényi network in Figure 3.8(c) shows many nodes with relatively high degree. In the scale-free network shown in Figure 3.8(d), we can see a higher degree node representing the long tail of the power law distribution.[2]

[2]The degree distribution and the average degree are approximate in the case of the scale-free network due to its small size.



(a) Full mesh

(b) k-regular

(c) Erdős-Rényi

(d) Scale-free

Figure 3.8: Examples of networks with several types of degree distributions.

3.4. Search Models

We define a *search* to be a walk in a network whose goal is to find a desired resource. Depending on the application, the resource looked for may be unique or multiple, i.e., it is held by a single node in the network or by multiple nodes. Resource multiplicity is taken care of in the analytical models. In this case, we define a *resource instance* to be any of the copies of a given resource. Resource uniqueness is clearly a particular case of resource multiplicity. In any case, our analysis considers that the presence of the desired resource in the network is guaranteed, i.e., the resource instances number at least one.

A search starts at a node that wants to locate the resource. This is called the *source* node. That search finishes when it visits a node that holds any instance of the desired resource. This is called a *target node*. Alternatively, a search may finish when it visits a neighbor of a node that holds the resource instance. This shortens search lengths at the expense of requiring each node to acquire and keep updated information about the resources held by its neighbors. This feature is called *one-hop resource replication* or just *one-hop replication*. In this case, the search finishes when it visits any neighbor of a target node. The models for some of the proposed mechanisms consider the possible presence of this feature in the network.

The number of hops taken by a search until it finds the resource is the *search length*. Since there is at least one instance of the resource in the network, the search length is finite. In practical settings where the existence of the resource can not be guaranteed, a stopping mechanism is needed to prevent searches from lingering using network resources indefinitely.

The main performance magnitude used to evaluate the performance of the resource location mechanisms is the *expected search length*. It has been chosen because it isolates the performance of the resource location itself, leaving out the impact of other factors such as node computing power and load, link bandwidth and traffic load, buffers lengths, etc. We define L to be a random variable representing the lengths of independent searches performed choosing source and target nodes uniformly at random in the network. The *expected search length* $\overline{L}$ is then defined as the expectation of this random variable. One of the main goals or our analysis will be to obtain analytical expressions for this magnitude.

3.5. Experimental Framework

The analytical models developed for the proposed resource location mechanisms are validated using experimental data. For this, an ad-hoc discrete-event simulator has been developed. The simulator capabilities include the generation and validation of random networks, the execution of independent searches, the collection of search data, and the elaboration of statistics on accumulated data. The simulator allows the specification of network and searches including the following parameters, which can be freely combined: the network size, its degree distribution, the networks construction method, the number of networks, the number of resource instances, the existence of one-hop replication, the type of random walks used, the use of partial walks, the length of partial walks, the number of partial walks per node, and the number of searches per network.

In accordance with what was mentioned in the previous section, one of our main goals will be to measure the lengths of the simulated searches and then to obtain the corresponding *average search length*. The details of how this average search length is obtained are specific to the experiments performed for each of the mechanisms and are deferred to the corresponding chapters. Average search lengths are used as real data to be compared with the expected search lengths predicted by the analytical models, and is the basis for validating those models. Where necessary, we use lower case to denote *average* search lengths ($\bar{l}$) and upper case for *expected* search lengths ($\overline{L}$).

Experiments have been performed for the three types of networks considered in this dissertation and introduced in the previous section: regular networks, Erdős-Rényi networks and scale-free networks. Network size has been set to $N = 10^4$ nodes in the base experiment, and several average degrees have been used to observe the effect of network connectivity. The number of networks considered and the number of searches per network vary with the considered mechanism, and are detailed in the corresponding chapters, along with the parameters specific to each algorithm.

Networks are randomly built by the simulator with the method proposed by Newman et al. [67] for networks with arbitrary degree distribution, setting their average node degree to $\overline{k} = 10$ in the base experiments. Each network is constructed in three steps:

1. A preliminary network is randomly constructed according to its size and a tentative degree distribution.

2. The actual degree distribution of the preliminary network is extracted.

3. The final network is obtained feeding the referenced method with that degree distribution.

This procedure and the use of the Newman's method achieves the intended network type (a feasible degree distribution that approximates the original input distribution), while guaranteeing that each attachment point is linked to any other attachment point in the network with equal probability, which is an assumption of the network model as stated in the previous section. Each generated network is checked for total connectivity before it is used to execute searches in it.

Chapter 4

Resource Location Based on Self-Avoiding Random Walks

This chapter presents the first resource location mechanism proposed in this dissertation to improve searches based on simple random walks in complex networks. It is, as the rest of the proposed mechanisms, targeted to unstructured systems, which were described in Section 2.2. This mechanism is based on the use of self-avoiding random walks, defined in Section 3.2.2, instead of simple random walks, defined in Section 3.2.1.

The model for the search mechanism is described in Section 4.1. An analysis for this model is developed in Section 4.2. Finally, the performance of the mechanism is evaluated in Section 4.3. The structure of this chapter is consistently used in Chapters 5 and 6 for the the rest of resource location mechanisms proposed in this dissertation.

4.1. Model

Let $G(V, E)$ be an undirected graph representing a communication network where V is the set of its nodes and E is the set of its links. The network is assumed to be connected, and it is characterized solely by the number of nodes N and by its degree distribution p_k. Each node has a number of attachment points equal to its degree, and the total number of attachment points in the network is $S = \sum_k kn_k$, which is also twice the total number of links in the network. Auto-links (links connecting a node to itself) and multilinks (more than one link between two nodes) are disallowed [37, 78].

The proposed model does not take into account the topology of G to characterize networks. This means that it is in fact a *mean-field model*, whose accuracy is checked using simulation experiments. Empirical results

for sets of regular, ER and scale-free networks built randomly show that model predictions are good approximations for those classes of networks. However, we believe it is possible to find classes of networks for which the structure is more important than the degree distribution, and for which the mean-field model predictions are not good approximations of real values.

The fact that the model does not consider network topology is reflected in the formalism in that it is assumed that given any attachment point of an arbitrary node i, the link departing from it can take us to any other attachment point (of a node $j \neq i$) in the network with equal probability. This condition, used also in the mean-field study in [78], is true for networks built with the random mechanism proposed by Newman [67].

We define a *self-avoiding walk* (SAW) as a random walk governed by the rules: (R_1) if all neighbors of the current node have been already visited, choose the next node uniformly at random among all neighbors (including the one the walker came from), and (R_2) if there is at least one unvisited neighbor of the current node, choose the next node uniformly at random among the unvisited neighbors.

We assume that the walker will search the network for a particular resource, starting at a node chosen uniformly at random from all network nodes (at hop $h = 0$). There are $R > 0$ instances of the resource sought. This allows us to model scenarios with multiplicity of resource instances such as peer-to-peer (P2P) networks, where several peers contain the same file [32]. We assume that these instances are held by (different) nodes chosen uniformly at random. In networks without one-hop replication, the search will finish when the walker visits any node holding an instance of the resource. In networks with one-hop replication, visiting a neighbor of any node holding the resource will finish the search.

4.2. Analysis

This section analyzes the model presented in the previous section. In particular, the analysis estimates the *network coverage* attained at each hop of the walk and uses this to estimate the *average search length*. We follow the approach that Rodero-Merino et al. [78] apply to simple random walks in one-hop replication networks, i.e., we derive recurrence formulas that estimate model magnitudes as functions of their values at the previous hop. Our model considers the availability or unavailability of one-hop replication in the network. It also considers the existence of multiple instances of the desired resource. Furthermore, the model in [78] has been extended to consider networks without one-hop replication and multiple resource instances

in searches based on simple random walks. The model developed in this dissertation is based on networks determined only by their size and degree distribution, with no particular topology assumed. Therefore, it is, as the one in [78], a mean-field model.

Our ultimate goal is to obtain an expression for the *expected search length* (denoted $\overline{L}_{saw}$), defined as the expectation of the number of hops it takes to find an instance of the resource sought. This magnitude depends on the number of *different* nodes that the walker has *visited* (denoted $V(h)$) or *covered* (denoted $C(h)$) up to and including hop h. In turn, $V(h)$ and $C(h)$ depend on a number of auxiliary magnitudes that capture the behavior of the SAW. These magnitudes are defined in Table 4.1 and their expressions are derived in the next paragraphs.

4.2.1. Visited Nodes

We start by estimating $P_w^k(h)$, the probability of visiting any node j of degree k at hop h, given that the walker is currently at some node i. Two different cases need to be considered: (1) all neighbors of i have already been visited, and (2) at least one neighbor of i has not been visited yet, corresponding to rules R_1 and R_2 of the SAW, respectively.

- Case 1: This probability is the ratio of "positive" attachment points and possible attachment points. We consider attachment points belonging to nodes of degree k to be positive and all attachment points in the network to be possible[1]. The probability of visiting any node j of degree k at hop h, conditioned on the fact that all neighbors of i have already been visited is then:

$$\begin{aligned} P_{w,anv}^k &= \frac{kn_k}{S} = \frac{kn_k}{\sum_j jn_j} = \frac{kp_k}{\sum_j jp_j} = \frac{kp_k}{\overline{k}}, \quad \text{for } h > 0, \\ P_{w,anv}^k(0) &= p_k. \end{aligned} \tag{4.1}$$

  Here, we are using our assumption that a given attachment point of i can take us to any attachment point in the network with equal probability. We are also taking into account the fact that the attachment

[1] To be fully precise, the attachment points of i should not be considered either as possible or positive (in case the degree of i is k), since auto-links are not allowed. However, that would introduce dependency on the degree of i, cluttering the analysis. Results for large randomly generated regular, ER and scale-free networks show that no significant error is introduced by this simplification.

Magnitude	Definition
$P_w^k(h)$	Probability of the walker arriving at any node of degree k at hop h, with $P_w^k(0) = p_k$.
$P_{w,anv}^k$	Probability of the walker arriving at any node of degree k at any hop, conditioned on that all the neighbors of the previous node have already been visited (*"anv" = all neighbors visited*).
$P_{w,nanv}^k(h)$	Probability of the walker arriving at any node of degree k at hop h, conditioned on that at least one neighbor of the previous node has not been visited yet (*"nanv" = not all neighbors visited*).
$P_{anv}^k(h)$	Probability that all the neighbors of the node reached at hop h have already been visited by the walker (*"anv" = all neighbors visited*), conditioned on that the degree of that node is k.
$P_{anv}(h)$	Probability that all the neighbors of the node (of any degree) reached at hop h have already been visited by the walker (*"anv" = all neighbors visited*).
$P_{nv}(h)$	Probability that a neighbor (any) of the node reached at hop h (excluding the previous node) has already been visited by the walker (*"nv" = neighbor visited*).

Table 4.1: Definitions of the auxiliary magnitudes of the model.

point of i will be chosen uniformly at random for hop h, as R_1 states. Note that this probability is not a function of the hop h (except for $h = 0$), since the behavior is that of a RW, whose next state depends only on the current state and not on the past history of the walk. For $h = 0$, the node is chosen uniformly at random among all the nodes in the network, so the probability of reaching a node of degree k is p_k.

- Case 2: Positive cases are now only those attachment points belonging to non-visited nodes of degree k. Similarly, possible attachment points are now only those belonging to all non-visited nodes. Let us denote by $V^k(h)$ the expected number of *different* nodes of each degree k visited by the walker up to and including hop h. We obtain the number of non-visited nodes from the number of visited nodes at the previous hop ($V^k(h-1)$). Thus, the probability of visiting any node j of degree k at hop h, conditioned on the fact that not all neighbors of i have already been visited is:

$$\begin{aligned} P^k_{w,nanv}(h) = \frac{k(n_k - V^k(h-1))}{S - \sum_j jV^j(h-1)} &= \frac{k(p_k - \frac{V^k(h-1)}{N})}{\overline{k} - \sum_j j\frac{V^j(h-1)}{N}}, \text{ for } h > 0, \\ P^k_{w,nanv}(0) &= p_k. \end{aligned} \tag{4.2}$$

This probability does depend on the hop h since the number of visited nodes increases as the walker proceeds through the network.

Using Equations 4.1 and 4.2, and taking into account that $P_{anv}(h-1)$ and $(1 - P_{anv}(h-1))$ are the probabilities that the SAW follows R_1 and R_2 at hop h, respectively, we have that $P^k_w(h)$ can now be obtained as:

$$\begin{aligned} P^k_w(h) = P^k_{w,anv} \cdot P_{anv}(h-1) &+ P^k_{w,nanv}(h) \cdot (1 - P_{anv}(h-1)), \text{ for } h > 0, \\ P^k_w(0) &= p_k. \end{aligned}$$

On the other hand, $P_{anv}(h)$ depends on the number of neighbors of the node reached at hop h (i.e., its degree), and can be obtained from the probabilities that all neighbors of the node have already been visited conditioned on the fact that the degree of that node is k:

$$P_{anv}(h) = \sum_k P^k_{anv}(h) \cdot P^k_w(h).$$

At this point, we note that:

$$P^k_{anv}(h) = (P_{nv}(h))^{k-1}.$$

The node from which the walk came from is excluded from the calculation (thus the exponent $k-1$), since it is certain that this node has already been visited. We are implicitly assuming here that each of the neighbors of the node are visited or not independently from the others. This approximation has been shown to be accurate in the types of network we consider [78].

The probability of a neighbor having been visited depends in turn on the degree of that neighbor, since nodes with higher degree have higher chance of being visited. This probability is thus calculated as:

$$\begin{aligned} P_{nv}(h) &= \sum_k P^k_{w,anv} \frac{V^k(h-2)}{n_k}, \text{ for } h > 1, \\ P_{nv}(0) &= 0, \\ P_{nv}(1) &= 0. \end{aligned}$$

$P^k_{w,anv}$ plays here the role of the probability that the neighbor is of degree k (the probability of visiting a node of degree k from the current node when it is chosen uniformly at random), while the fraction within the summation gives the probability of a neighbor of degree k being already visited as the ratio of positive nodes and possible nodes. Here, $V^k(h-2)$ is used instead of $V^k(h)$ to exclude the node reached at hop h (in case its degree is k) and the node the walk came from (in case its degree is k). The latter is certain to have been visited already as stated above, and thus does not participate in the calculation of $P^k_{anv}(h)$.

$V^k(h)$ can now be estimated using the following recursive expression:

$$\begin{aligned} V^k(h) &= V^k(h-1) + P^k_{w,nanv}(h) \cdot (1 - P_{anv}(h-1)), \text{ for } h > 0, \\ V^k(0) &= p_k, \end{aligned}$$

since a new node of degree k is visited at hop h with probability $P^k_{w,nanv}(h)$ given that the routing algorithm follows R_2, which in turn happens with probability $(1 - P_{anv}(h-1))$. The expected total number of different nodes visited by the walk up to and including hop h is then:

$$V(h) = \sum_k V^k(h).$$

4.2.2. Covered Nodes

This magnitude estimates the expected number of nodes covered by the walker up to and including hop h, denoted by $C(h)$. A node is *covered* by the SAW when the walker visits that node or any one of its neighbors. The number of nodes covered by the walker allows the estimation of average search length in one-hop replication networks in the last step of our analysis, since to find a resource in such a network it suffices to *cover* (not to *visit*) the node that holds it.

We first estimate $C^k(h)$, the average number of different nodes of degree k covered by the walker up to and including hop h, in order to obtain $C(h)$. For networks with a given degree distribution, the average number of nodes of degree k that a walker covers at each hop clearly depends on the routing algorithm used. An expression for $C(h)$ was obtained in [78] for a simple RW in the same types of network as those considered in our work. It is expressed in terms of the expected number of visited nodes of each degree ($V^k(h)$) and the degree distribution of the network, where $V^k(h)$ contains the effect of the routing algorithm. Therefore, that expression is still valid if we change the random walk into a SAW, provided that we use the expressions for $V^k(h)$ derived above for the SAW. Thus, the expected number of covered nodes of degree k can be written as:

$$\begin{aligned} C^k(h) &= C^k(h-1) + \left(\frac{k\left(n_k - C^k(h-1)\right)}{S - \sum_j j \cdot V^j(h-1)} \right) \cdot \\ &\quad \sum_j \left(V^j(h) - V^j(h-1)\right)(j-1), \quad \text{for } h > 0, \\ C^k(0) &= V^k(0) + \overline{k} P^k_{w,anv}. \end{aligned}$$

Finally, the expected number of covered nodes is obtained as:

$$C(h) = \sum_k C^k(h).$$

4.2.3. Search Length

The expected search length is obtained from the probability of finishing the search at hop h ($P_{fin}(h)$), that is in turn obtained from the probability of being successful in finding the resource at hop h ($P_{suc}(h)$):

$$\overline{L}_{saw} = \sum_{h=0}^{\infty} h \cdot P_{fin}(h), \tag{4.3}$$

where

$$\begin{aligned} P_{fin}(h) &= P_{suc}(h)\prod_{i=0}^{h-1}(1-P_{suc}(i)), \text{ for } h>0, \\ P_{fin}(0) &= \frac{R}{N}. \end{aligned}$$

Recall that there are R instances of the resource, placed in different nodes. Now, $P_{suc}(h)$ depends on whether or not the networks considered are one-hop replication networks. For networks without replication, $P_{suc}(h)$ can be estimated as the relation between the number of new nodes visited at hop h and the number of nodes that are still unvisited at that hop, taking into account the number of resource instances:

$$\begin{aligned} P_{suc}(h) &= R\frac{V(h)-V(h-1)}{N-V(h-1)}, \text{ for } h>0, \\ P_{suc}(0) &= \frac{R}{N}. \end{aligned} \tag{4.4}$$

For one-hop replication networks, $P_{suc}(h)$ can be similarly estimated as the relation between the number of new nodes covered at hop h and the number of nodes that are still uncovered at that hop:

$$\begin{aligned} P_{suc}(h) &= R\frac{C(h)-C(h-1)}{N-C(h-1)}, \text{ for } h>0, \\ P_{suc}(0) &= \frac{R}{N}. \end{aligned} \tag{4.5}$$

From Equations 4.3, 4.4 and 4.5 we can finally write the expected search length in networks with no replication ($\overline{L}_{saw,nrep}$) and in one-hop replication networks ($\overline{L}_{saw,rep}$), respectively, as:

$$\begin{aligned} \overline{L}_{saw,nrep} = \sum_{h=1}^{\infty} h \Bigg[& R\frac{V(h)-V(h-1)}{N-V(h-1)} \cdot \left(1-R\frac{V(0)}{N}\right) \cdot \\ & \prod_{i=1}^{h-1}\left(1-R\frac{V(i)-V(i-1)}{N-V(i-1)}\right)\Bigg], \end{aligned}$$

and

$$\begin{aligned} \overline{L}_{saw,rep} = \sum_{h=1}^{\infty} h \Bigg[& R\frac{C(h)-C(h-1)}{N-C(h-1)} \cdot \left(1-R\frac{C(0)}{N}\right) \cdot \\ & \prod_{i=1}^{h-1}\left(1-R\frac{C(i)-C(i-1)}{N-C(i-1)}\right)\Bigg]. \end{aligned}$$

Note that both $\overline{L}_{saw,nrep}$ and $\overline{L}_{saw,rep}$ depend only on the network parameters, namely N, p_k and R.

4.3. Performance Evaluation

In this section, we assess the accuracy of the SAW model comparing its predictions against simulation results. The experimental framework described in Section 3.5 is used. At the same time, we compare the performance of SAWs and simple RWs for searching complex networks. Experiments have been performed for three types of randomly built networks: regular networks, Erdős-Rényi networks (ER) and scale-free networks (see Section 3.3.4). Network size has been set to $N = 10^4$ nodes in the base experiment, and three average degrees have been used ($\overline{k} = 10, 20, 30$) to observe the effect of network connectivity.

ER networks with degree averages $\overline{k} = 10, 20, 30$ have been obtained with link probabilities $p = 0.001, 0.002$ and 0.003, respectively. Scale-free networks follow a power-law degree distribution ($p(k) \propto k^{-\gamma}$) with γ values of 2.7030, 1.9958 and 1.781, adjusted to obtain feasible networks with average degrees $\overline{k} = 10, 20, 30$, respectively. For this type of network, a minimum degree of 5 has been set to avoid disconnected networks due to the high numbers of low degree nodes. The degree distributions thus obtained are then used to feed the SAW and RW models[2], and to construct the networks for simulations using Newman's method. This method guarantees that each attachment point is linked to any other attachment point in the network with equal probability, which is an assumption of the network model as stated in Section 3.3.3.

To be able to fairly compare the predictions of the mean-field models (*expected* values) with the experimental results (*average* values)[3], simulations have been run for a representative number of different networks (10^2) with that degree distribution, and for a representative number of searches (10^3) for each of those networks, resulting in a total of 10^5 simulated searches for each experiment. Magnitudes are measured for each search and values averaged over all searches in that network to obtain the *network average*. Network averages are in turn averaged over all networks to obtain the total

[2]For simple RWs, we have used the analytical model in one-hop replication networks proposed in [78], with modifications to include networks without one-hop replication and several instances of resources.

[3]Throughout this section, for conciseness reasons, we sometimes omit the qualifications *expected* and *average*. For instance, when we are showing both the *expected search lengths* from an analytical model and the *average search lengths* from a simulation experiment, we refer to them just as the *search lengths*.

average values to be compared with those the expected values estimated by the models. Variations of network averages compared to total average values have also been studied, finding that average values for individual networks are close to total network average values. Results for minimum, average, and maximum values, along with the standard deviation, are shown at the end of this section.

We begin by showing results for two auxiliary magnitudes: the probability that all neighbors of a node have already been visited and the probability of visiting a node of degree k. These results will be shown for ER networks only, as an illustration of the behavior of the SAW. Next, we show results for the number of nodes visited $V(h)$ and covered $C(h)$ by the walkers, which are determined by the previous probabilities. Finally, results for search lengths ($\overline{L}_{saw}$ and $\overline{L}_{rw}$) are shown.

4.3.1. Auxiliary Magnitudes

Figure 4.1 shows $P_{anv}(h)$, the probability that all the neighbors of a node (of any degree) reached at hop h have already been visited, in ER networks with $\overline{k} = 10$. It is an interesting starting point, since this probability can also be interpreted as the probability that the SAW actually succeeds in avoiding already visited nodes, or else it falls back to the RW behavior. In this and following figures, *expected* data from analytical models are shown as curves, and *average* data from simulations are shown as points.

This figure clearly shows three different regions or phases, which correspond to different behaviors of the SAW. The probability that all neighbors of the node have already been visited is very low up to around 7500 hops, which is 75% the size of the network. Therefore, the SAW will visit a new node in every hop with high probability in this region. Then, the probability that all neighbors have already been visited grows rapidly in the central region of the graph (roughly between 7500 and 12500 hops). This is a phase of transition between the SAW behavior and the RW behavior. For numbers of hops greater than 12500, the probability grows asymptotically to 1. In this region the SAW behaves very much like a RW, since it is almost certain that all neighbors of the nodes reached have already been visited.

Figure 4.1 also shows that the model predicts results with a reasonable accuracy. The more significant deviations between model and simulation results occur in the central region of the graph (transition phase). For smaller number of hops, the model is *pessimistic*, that is, it predicts higher probabilities that all neighbors are visited than the simulation does. This is inverted for greater number of hops. This will yield pessimistic estimations of the average number of visited and covered nodes, as will be shown in later para-

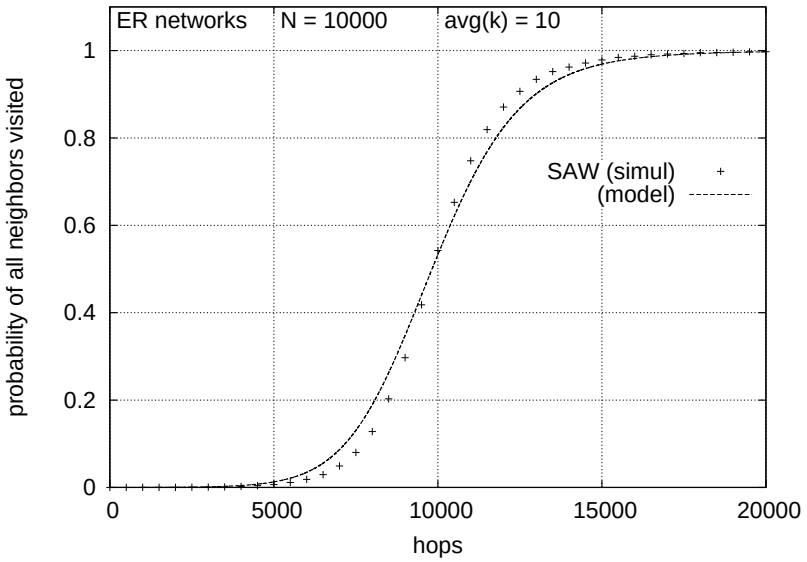


Figure 4.1: Probability of all neighbors visited ($P_{anv}(h)$) by SAW in ER networks with $\overline{k} = 10$.

graphs. Therefore, the analysis presented in Section 4.2 is a *conservative* model of the real SAW behavior.

Figure 4.2 shows $P_w^k(h)$, the probability of visiting a node of degree k at hop h. This magnitude is interesting because it shows, like the previous one, the different phases of the algorithm. In addition, $P_w^k(h)$ is the base to estimate the average number of visited and covered nodes, which in turn are used to estimate the average search length. Curves are presented for three representative degrees: the average degree ($k = \overline{k} = 10$), a degree above the average ($k = 15$), and a degree below the average ($k = 5$). The first thing to note about this graph is that the probabilities for each degree start (low h) and finish (high h) at the same values. Indeed, when very few nodes are visited, the probability of reaching a node of a given degree is very similar to that probability of the RW model, since the next hop is chosen uniformly at random among the (non-visited) neighbors of the node. Likewise, when almost all nodes have been visited, the probability of visiting a node of some degree is very similar to that of the RW, since the next hop is chosen uniformly at random among the (already visited) neighbors of the node. The probability of arriving at a node of degree k at any hop (P_A^k) in the RW model [78] is the same as the probability of visiting a node of degree k at any hop if all the neighbors of the current node have already been visited ($P_{w,anv}^k$) in the SAW model. Recall from Section 4.2.1 that this probability

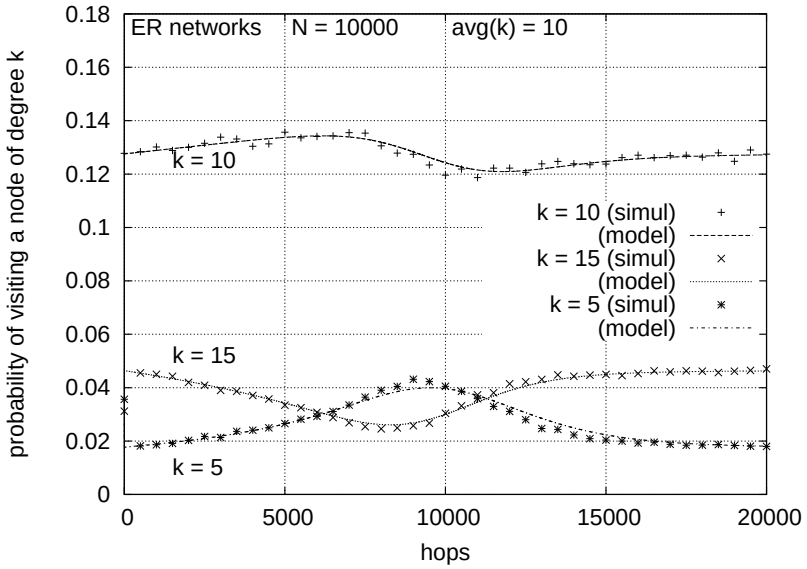


Figure 4.2: Probability of visiting a node of degree k ($P_w^k(h)$) by SAW in ER networks with $\overline{k} = 10$.

is not a function of h, since RW is a memory-less algorithm. Therefore, in the SAW model $P_w^k(h)$ starts at $P_{w,anv}^k = \frac{kp_k}{\overline{k}}$ for low h and tends to the same value for high h.

In the first section of the graph (up to about 7500 hops), the SAW visits a new node at each hop with high probability. Since nodes with higher degrees are visited faster, the probability of visiting more new nodes of those degrees decrease as h increases. This effect can be seen in Figure 4.2 in the curve for $k = 15$. Nodes with lower degrees are visited more slowly, so the probability of visiting more new nodes of those degrees increases with h. This effect can also be seen in curves for $k = 5$ and $k = 10$.

In the second region of the graph (between 7500 and 12500 hops), the transition phase in Figure 4.1, the probability that all neighbors have been already visited grows rapidly. This will make the probability of reaching nodes of higher degrees grow to recover the level of $P_{w,anv}^k$ in the third region of the graph ($h > 12500$). On the other hand, the probability of reaching nodes of lower degrees falls again to the level of $P_{w,anv}^k$.

4.3.2. Visited Nodes

Figure 4.3 shows the expected and average number of visited nodes as a function of the number of hops taken by the walker ($V(h)$) for the three

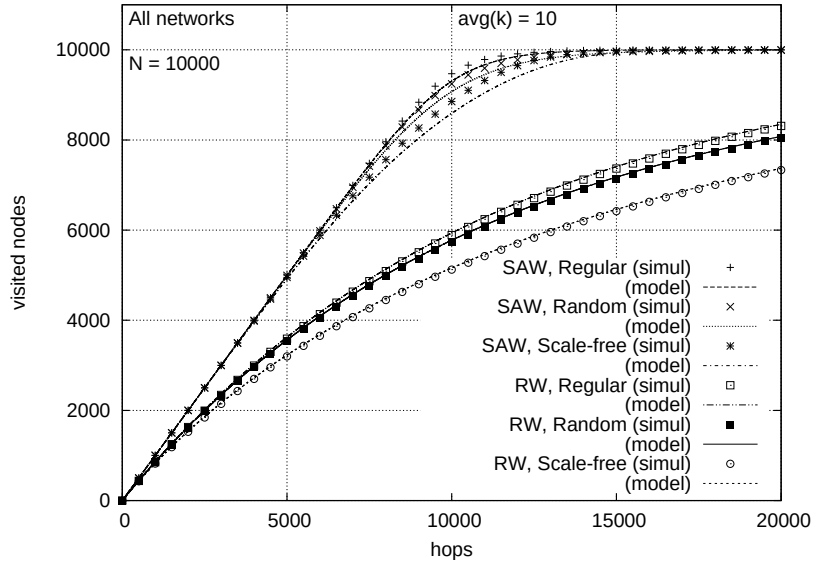
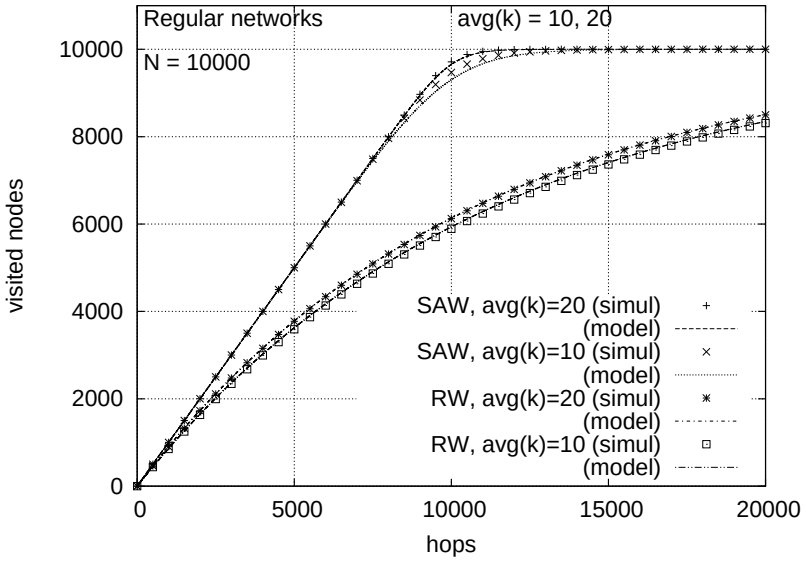


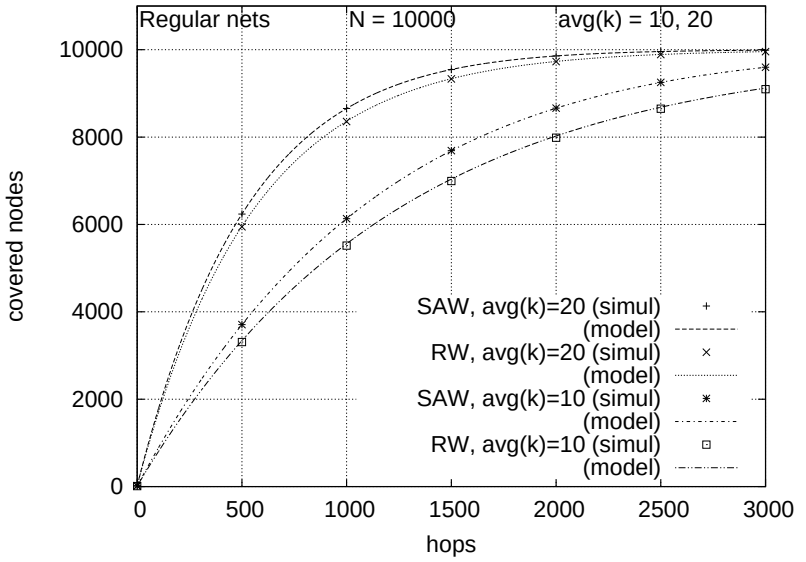
Figure 4.3: Number of visited nodes of all degrees ($V(h)$) in the three network types, for $\overline{k} = 10$.

types of networks with $\overline{k} = 10$. As expected, curves for RW grow more slowly than curves for SAW, since the latter ones try to avoid revisiting nodes thus visiting new nodes faster than the former ones. In fact, we notice that SAW achieves a straight line with slope 1 in the first phase of the algorithm, that is, it visits a new node at each hop with a high probability up to a number of hops about 75% of the network size. From then on, new nodes are visited more slowly in the second phase until the whole network is covered in the third phase. The graph also shows that at least 99% of the network has been visited by the SAW when only (approximately) 75% has been visited by the RW.

Both SAW and RW curves are higher for regular networks. Curves for ER networks are slightly lower and curves for scale-free networks are still farther down. This is an effect of the different degree distributions of the three types of networks. In regular networks, all nodes are visited with the same probability, since all of them have the same degree. In random and scale-free networks a number of different degrees are present in the network. Nodes with smaller degrees are visited with smaller probability than nodes with larger degrees. It is harder for both SAW and RW to visit nodes with small degrees, resulting in a higher number of revisited nodes and thus in a slower rate of visiting new nodes. This effect is greater in scale-free networks due to the shape of power-law distributions: a large number of small degree nodes

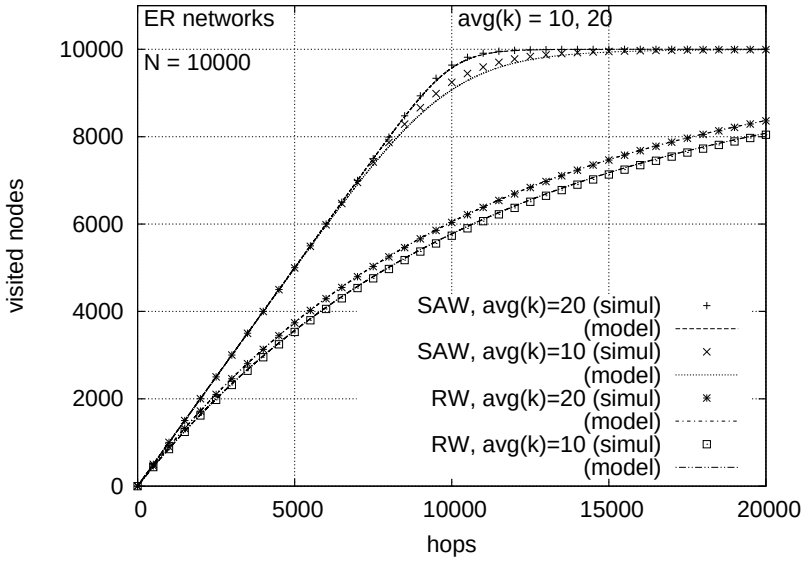


(a) Visited nodes

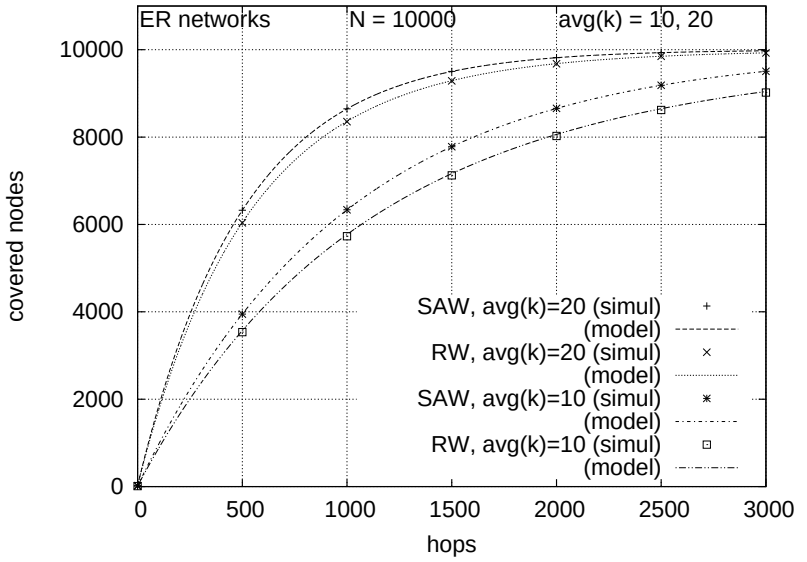


(b) Covered nodes

Figure 4.4: Number of visited ($V(h)$) and covered ($C(h)$) nodes of all degrees in regular networks, for $\overline{k} = 10, 20$.

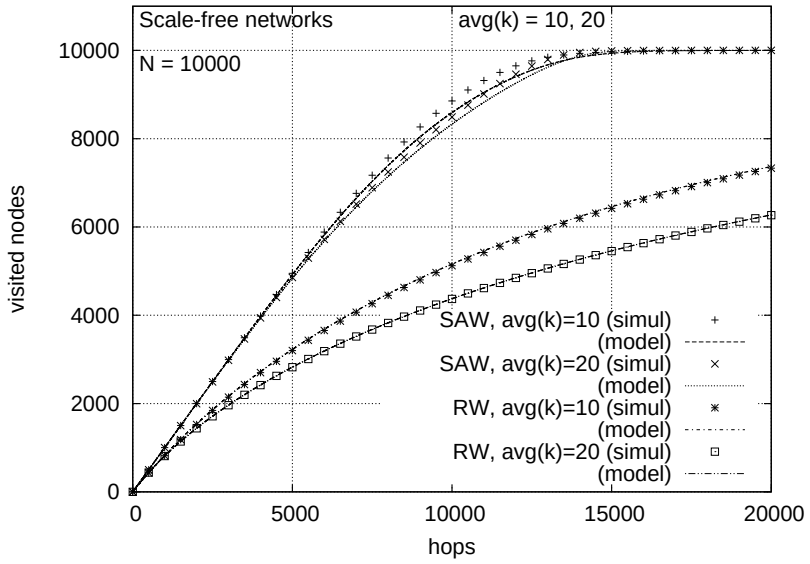


(a) Visited nodes

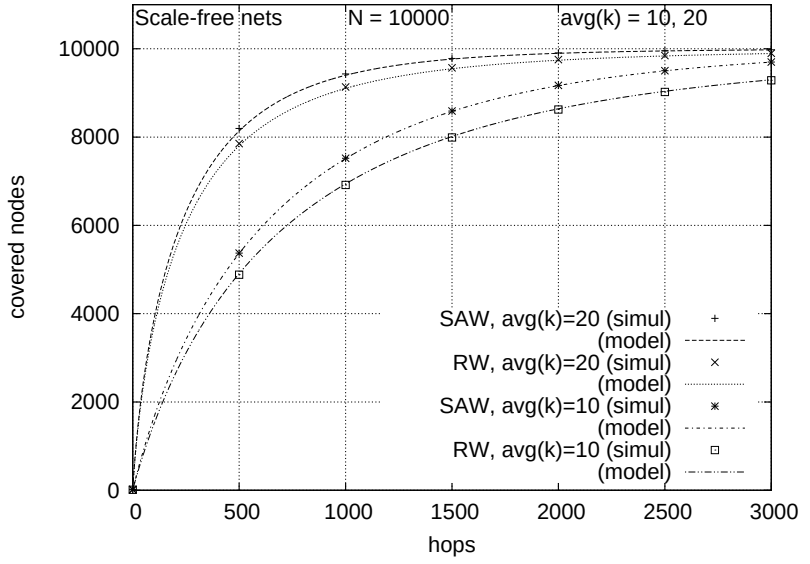


(b) Covered nodes

Figure 4.5: Number of visited ($V(h)$) and covered ($C(h)$) nodes of all degrees in ER networks, for $\overline{k} = 10, 20$.



(a) Visited nodes



(b) Covered nodes

Figure 4.6: Number of visited ($V(h)$) and covered ($C(h)$) nodes of all degrees in scale-free networks, for $\overline{k} = 10, 20$.

and a few nodes with a very high degree (the long tail of the distribution).

Now, we compare the average number of visited nodes in networks with average degrees $\overline{k} = 10$ and $\overline{k} = 20$ in Figures 4.4(a), 4.5(a) and 4.6(a) (regular, ER, scale-free networks, respectively). For regular and ER networks, curves for $\overline{k} = 20$ are always higher than those for $\overline{k} = 10$ for both SAW and RW. Since nodes in higher average degree networks tend to have higher degrees, it is easier to visit new nodes either randomly (RW) or trying to avoid already visited nodes (SAW), and thus the faster rate of visiting new nodes. However, the effect is reversed for scale-free networks: curves for $\overline{k} = 20$ are lower than those for $\overline{k} = 10$. This can be explained again by the shape of the power-law degree distribution. A higher average degree network has more very high degree nodes than a lower average degree network. Therefore, walks keep visiting these nodes with higher probability, making it more difficult to visit low degree nodes. It can also be seen that the difference is larger for RW than for SAW, since the latter tries to avoid high degree nodes once they have been visited for the first time, increasing the probability of visiting new nodes.

To further investigate this behavior in the case of SAW, we have checked the probability that all neighbors of a node have been already visited, since this is also the probability that the next node will not be a new one. In networks with higher average degree, this probability is indeed higher up to a number of hops close to the network size. This accounts for the growing divergence between curves for $\overline{k} = 10$ and $\overline{k} = 20$ in Figure 4.6(a). For subsequent hops, the probability of all neighbors being visited is higher in networks with lower average degree, explaining the convergence of the curves as they grow towards a number of visited nodes equal to the network size. The behavior of the probability of all neighbors being visited for SAW in regular and ER networks has been checked to be opposite to that of scale-free networks, consistent with the observed opposite behavior of the number of visited nodes.

4.3.3. Covered Nodes

Figure 4.7 shows the expected and average number of covered nodes as a function of the number of hops taken by the walker ($C(h)$) for networks of the three types with $\overline{k} = 10$. Although SAW always achieves better rates than RW as expected, the difference here is much smaller than for the average number of visited nodes (Figure 4.3). Although RW visits fewer new nodes than SAW, it covers almost the same number of nodes. Although more work needs to be done to find out the reason for this, it appears that the neighboring relations among nodes partially compensate the extra randomness of

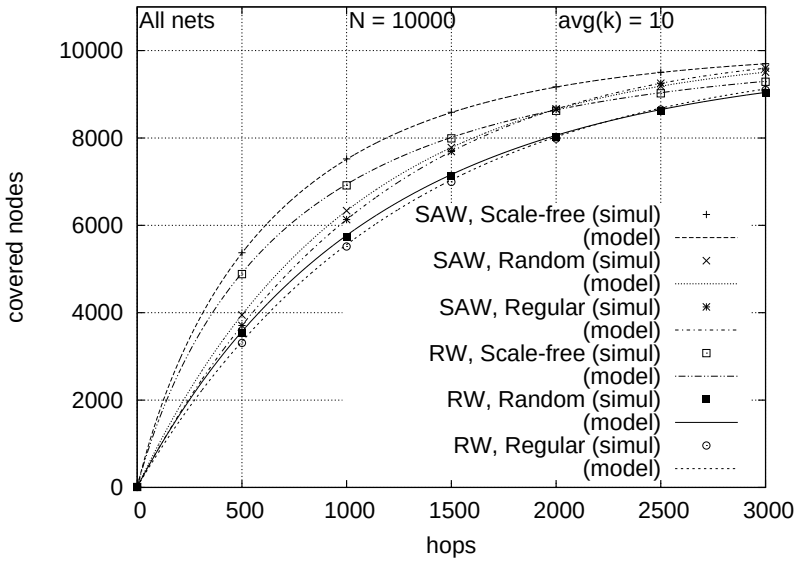


Figure 4.7: Number of covered nodes of all degrees in the three network types, for $\overline{k} = 10$.

RW with respect to SAW. This effect reduces the difference in the performance of both algorithms when applied to searching networks with one-hop replication, as will be shown later in this section.

Figures 4.4(b), 4.5(b) and 4.6(b) (regular, ER, scale-free networks, respectively) compare the expected and average number of covered nodes in networks with average degrees $\overline{k} = 10$ and $\overline{k} = 20$. In regular and ER networks, curves for $\overline{k} = 20$ are higher than those for $\overline{k} = 10$, the same as observed for the number of visited nodes in Figures 4.4(a) and 4.5(a). In scale-free networks, the curve for $\overline{k} = 20$ is also higher than that for $\overline{k} = 10$, in contrast with what was observed for the number of visited nodes (Figure 4.6(a)). Although visiting new nodes in scale-free networks is harder for those with higher average degree, it is easier to cover them because with high probability they are neighbors of the (more numerous) high degree nodes.

4.3.4. Search Length

This section shows results for the expected and average search length achieved by SAW ($\overline{L}_{saw}$) and RW ($\overline{L}_{rw}$) in the three types of networks, with and without one-hop replication. Recall from the definition of the model (Section 4.2.3) that $\overline{L}_{saw}$ is obtained from the average number of visited nodes ($V(h)$) for networks without one-hop replication, while it is obtained

from the average number of covered nodes ($C(h)$) for networks with one-hop replication. We begin by comparing results for a single instance of the resource in the three types of networks. Then, we study the dependency of $\overline{L}_{saw}$ on the number of instances of the resource in Section 4.3.5.

Figure 4.8 shows the average search length for SAW and RW in the three types of networks, with average degrees $\overline{k} = 10, 20, 30$, with and without one-hop replication. Model predictions (bars) are in good agreement with simulation results (points). The SAW model registers errors with respect to the simulations smaller than 1.2%, 1.6% and 4.9% in regular, ER and scale-free networks, respectively. More detail of these deviations are given in Table 4.3.

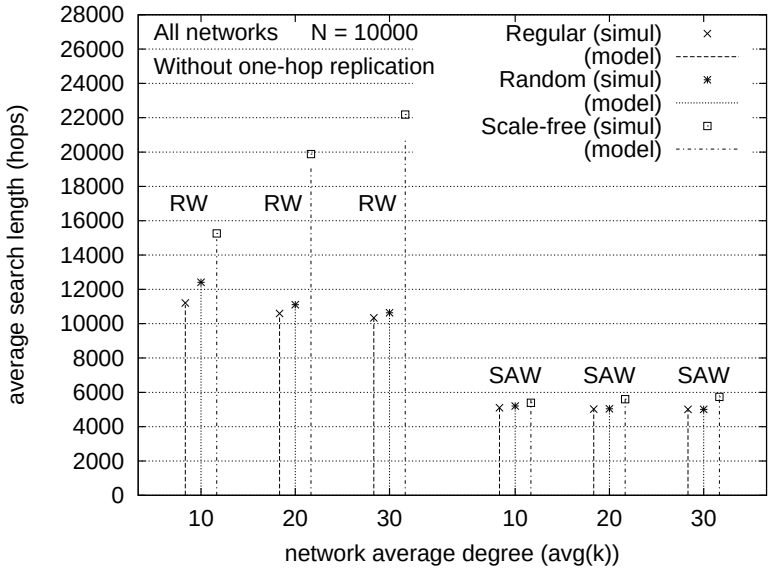
Dependency on One-Hop Replication

If we first pay attention to the absolute values of $\overline{L}_{saw}$ in networks with $\overline{k} = 10$, we notice that it is a little over half the network size in networks without one-hop replication, while it is around 10% the network size when the network has this feature. The former value agrees with Figure 4.3, where we observe that the SAW visits a new node at each hop most of the time, since the node that holds the instance of the resource sought has been randomly chosen. Likewise, the latter value agrees with Figure 4.7, where we observe that the number of nodes covered by the SAW at hop $h = 1000$ is about half of the network size.

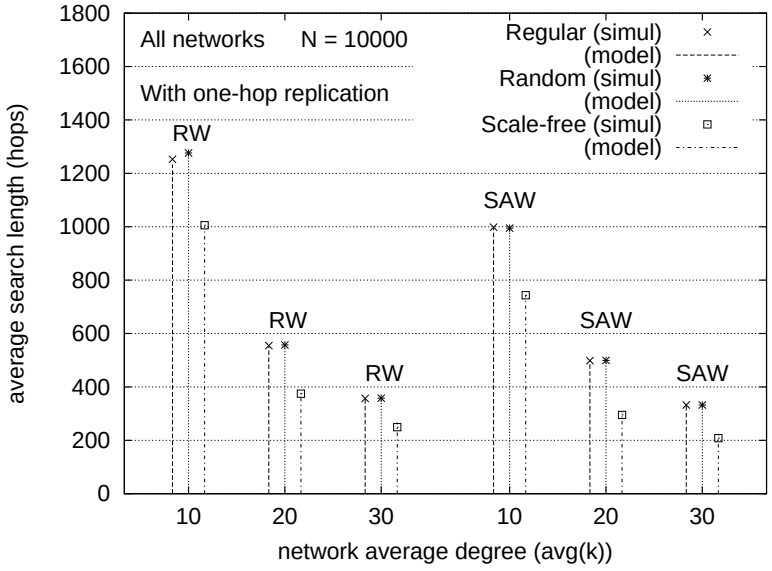
These graphs show two trivial results: $\overline{L}_{saw}$ is smaller than $\overline{L}_{rw}$ in the three types of networks; it is also smaller in networks with one-hop replication than in networks without this feature. It is more interesting to quantify the reduction in the average search length achieved by SAW with respect to RW for each network type. This information is presented in Table 4.2, where the reduction in search length is given as $(\overline{L}_{rw} - \overline{L}_{saw})/\overline{L}_{rw} \cdot 100(\%)$. For networks without one-hop replication, the reduction is above 50%, whereas for networks with this feature the reduction is smaller (above 20%). This result is consistent with what we obtained in the two previous sections for the number of visited and covered nodes. There (see Figures 4.3 and 4.7), we observed that there was a significant difference in curves of $V(h)$ for SAW and RW, while curves for $C(h)$ were almost coincident. This explains the smaller reductions of $\overline{L}_{saw}$ in networks with one-hop replication.

Dependency on the Network Type

Going back to Figure 4.8, if we pay attention to the comparison among the three types of networks, we observe that both SAW and RW show dif-



(a) Without one-hop replication.



(b) With one-hop replication.

Figure 4.8: Search lengths for SAW ($\overline{L}_{saw}$) and RW ($\overline{L}_{rw}$) with a single resource instance in the three types of networks, with and without one-hop replication, and for several degree averages.

		Reduction in $\overline{L}_{saw}$ (%)		
Network type	One-hop repl.	$\overline{k} = 10$	$\overline{k} = 20$	$\overline{k} = 30$
Regular	no	54.46	52.57	51.57
	yes	20.32	10.10	6.95
ER	no	57.92	54.60	52.89
	yes	22.47	10.40	7.17
Scale-free	no	64.67	71.82	74.16
	yes	26.07	21.31	19.88

Table 4.2: Reduction of the average search length achieved by SAW with respect to RW.

ferent effects in networks with and without one-hop replication. In networks without one-hop replication, the algorithms achieve values of $\overline{L}_{saw}$ in increasing order for regular, ER and scale-free networks. The largest increment is registered for RW in scale-free networks. This is due to the existence of a large number of small degree nodes in scale-free networks. These nodes are more difficult to visit by the walks, especially by RW, yielding larger search lengths. This negative effect is almost totally compensated by SAW, since it tries to avoid already visited nodes, incrementing the probability of visiting low degree nodes. A consequence of this is the fact that SAW gets the largest reduction of search length compared to RW in scale-free networks, as seen from Table 4.2.

In networks with one-hop replication, values of $\overline{L}_{saw}$ are similar for regular and ER networks. Scale-free networks present smaller values, as opposed to what happened in networks without one-hop replication. This is explained by the presence of very large degree nodes in scale-free networks. Although these nodes are few, they are visited with high probability, allowing many nodes to be covered without being visited, leading to reduced search lengths. This feature also yields larger reductions of $\overline{L}_{saw}$ in scale-free networks, as verified from Table 4.2.

Dependency on the Average Degree

We focus again on Figure 4.8 to analyze the dependency of $\overline{L}_{saw}$ on the average degree of the networks. For regular and ER networks, a larger $\overline{k}$ yields a smaller $\overline{L}_{saw}$ and $\overline{L}_{rw}$. For the former, the decrement is explained by the fact that the higher the degree of a node, the more probable it is to visit an unvisited neighbor of that node in the next hop. For the latter, the reduction comes from the fact that the probability that all the neighbors of

a node have already been visited is lower if the degree of the node is higher. In networks without one-hop replication, the reduction when the average degree increases is small for $\overline{L}_{rw}$ and irrelevant for $\overline{L}_{saw}$. The reduction of search length achieved by SAW with respect to RW slowly decreases with $\overline{k}$ (Table 4.2). In networks with one-hop replication, however, $\overline{L}_{saw}$ and $\overline{L}_{rw}$ are reduced by about one half when $\overline{k}$ is changed from 10 to 20, and about one third when $\overline{k}$ is changed from 20 to 30. The higher degree of the nodes allows walkers to cover nodes faster, since nodes know about more neighbors. The reduction of $\overline{L}_{saw}$ with respect to $\overline{L}_{rw}$ decreases faster with $\overline{k}$ in this case than in networks without one-hop replication (Table 4.2).

The impact of the average degree in scale-free networks with one-hop replication is similar to that described for regular and ER networks. However, the effect is reversed in networks without one-hop replication, where $\overline{L}_{saw}$ and $\overline{L}_{rw}$ grow for larger $\overline{k}$. For RW, the search length increases about 30% when $\overline{k}$ is changed from 10 to 20, and about 10% when $\overline{k}$ is changed from 20 to 30. The increment is less significant in SAW (under 5%). This increment is again due to the larger number of high degree nodes (visited with high probability) in a network with higher $\overline{k}$. This behavior makes SAW achieve a reduction of search length with respect to RW that grows with the average degree of the network, reaching 74% for $\overline{k} = 30$ (Table 4.2).

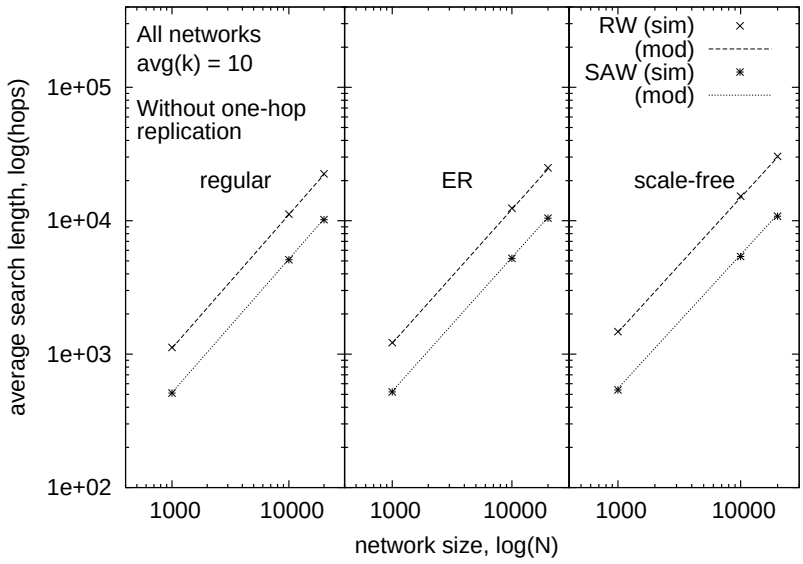
Dependency on the Network Size

Finally, we analyze the dependency of the expected and average search lengths on the size of the network. Simulation experiments for networks of $N = 1000$ and $N = 20000$ nodes have been added to the base experiments for $N = 10000$ nodes presented so far. Figure 4.9 show the average search length obtained for regular, ER and scale-free networks as a function of their size. The average degree of all networks is $\overline{k} = 10$.

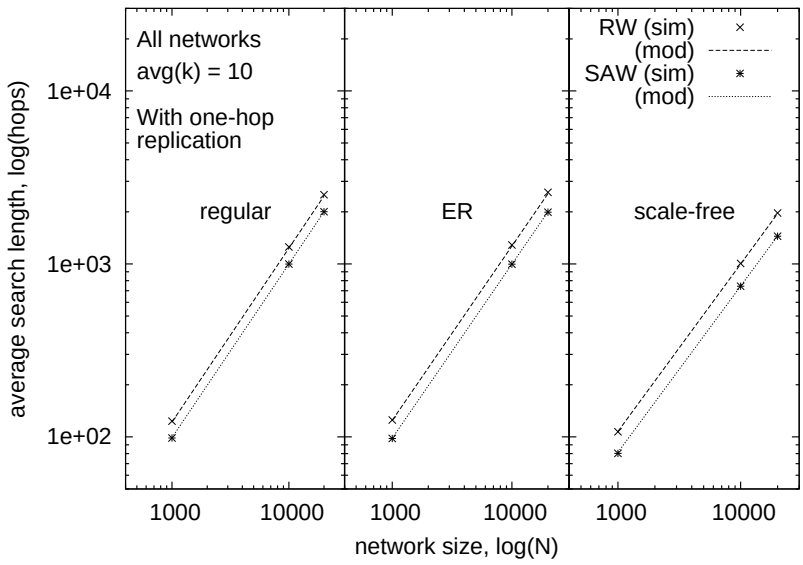
It is observed that the search length is linear in the network size when log scales are used. As for the previous experiments, model predictions are in good agreement with simulation results. The magnitude of deviations depends on the network type, on the presence of the one-hop replication feature and on the network size. Table 4.3 presents these deviations relative to average search lengths registered in simulations.

4.3.5. Resource Multiplicity

We now look at the dependency of the expected and average search lengths on the number of instances of the resource sought present in the network (R). Figures 4.10, 4.11, and 4.12 show this dependency for regular,



(a) Without one-hop replication.



(b) With one-hop replication.

Figure 4.9: Search lengths for SAW ($\overline{L}_{saw}$) and RW ($\overline{L}_{rw}$) for the three types of networks as a function of network size.

		$\overline{L}_{saw}$ model prediction errors (%)		
Network type	One-hop repl.	$N = 1000$	$N = 10000$	$N = 20000$
Regular	no	0.88	1.17	1.35
	yes	0.38	0.09	0.15
ER	no	1.29	1.57	1.51
	yes	0.53	0.06	0.44
Scale-free	no	2.72	3.20	3.07
	yes	1.37	0.09	0.36

Table 4.3: Relative errors of expected search lengths predicted by the model with respect to average lengths from simulations, for networks with $\overline{k} = 10$.

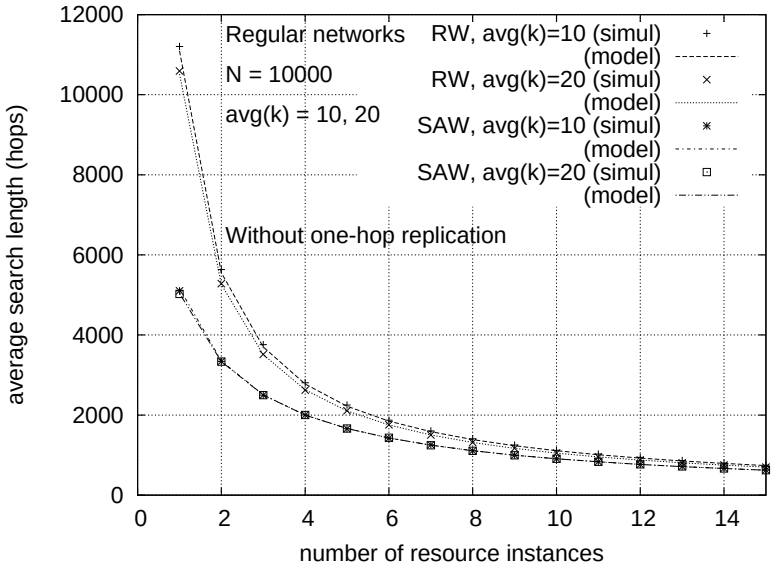
ER and scale-free networks, respectively, with and without one-hop replication, and for average degrees $\overline{k} = 10, 20$. Note the different scale of the y-axis in graphs labeled (a) without one-hop replication and (b) with one-hop replication.

We observe that the reductions in $\overline{L}_{saw}$ and $\overline{L}_{rw}$ are large for the first additional resource instances, asymptotically tending to 0 as R grows towards the network size. (In particular, we have checked that for SAW in networks without one-hop replication, this dependency is $\overline{L}_{saw} \approx N/(R+1)$.) In networks without one-hop replication, the difference in search length achieved by SAW and RW quickly diminishes with the number of resource instances. A similar behavior can be observed in networks with one-hop replication and with average degrees 10 and 20.

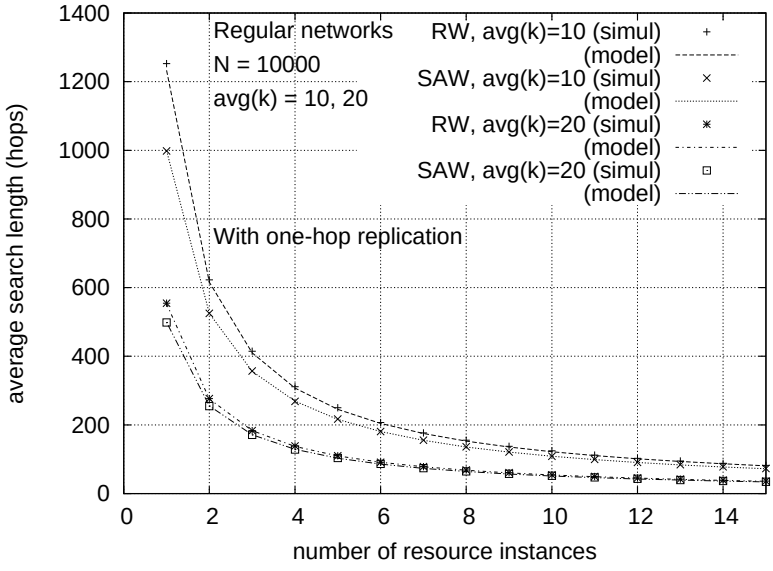
In these experiments, SAW searches outperform RW searches in networks without one-hop replication regardless of their average degree. This is not true for networks with one-hop replication, where an increase in $\overline{k}$ has a large impact on the rate at which the network is covered. This results in a better performance of RW in networks with $\overline{k} = 20$ than that of SAW in networks with $\overline{k} = 10$.

4.3.6. Variations of Network Averages

As stated in Section 4.1, our model of SAW is a mean-field analysis that produces an estimation of the expected search length in networks with a given size and degree distribution. Of course, individual walks on a given network can yield large deviations from the predicted expected value. To ensure the usefulness of the model when applied to an individual network (with the given degree distribution), the question that arises is now whether the choice of that particular topology can produce significant deviations. To

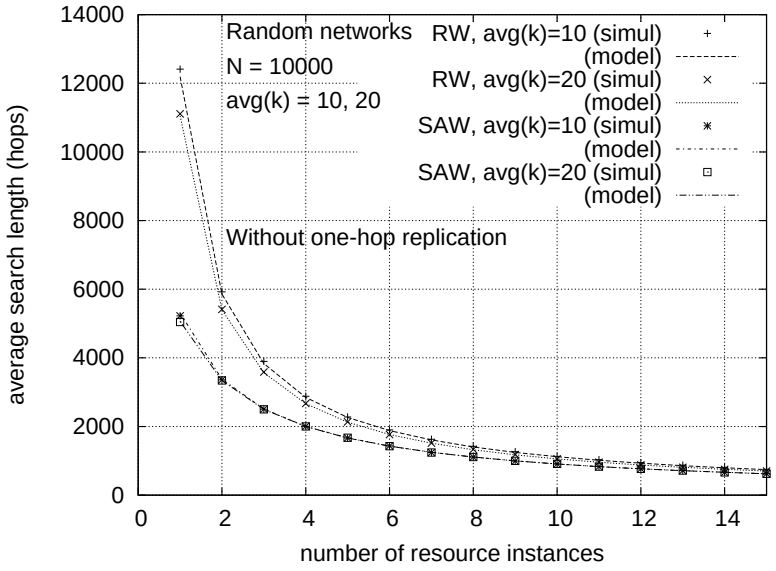


(a) Without one-hop replication.

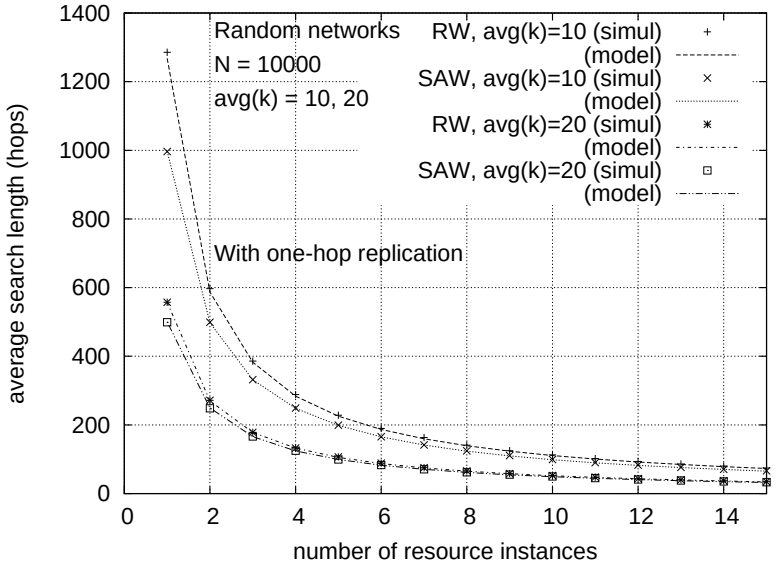


(b) With one-hop replication.

Figure 4.10: Search lengths for several resource instances in regular networks with $\overline{k} = 10, 20$.

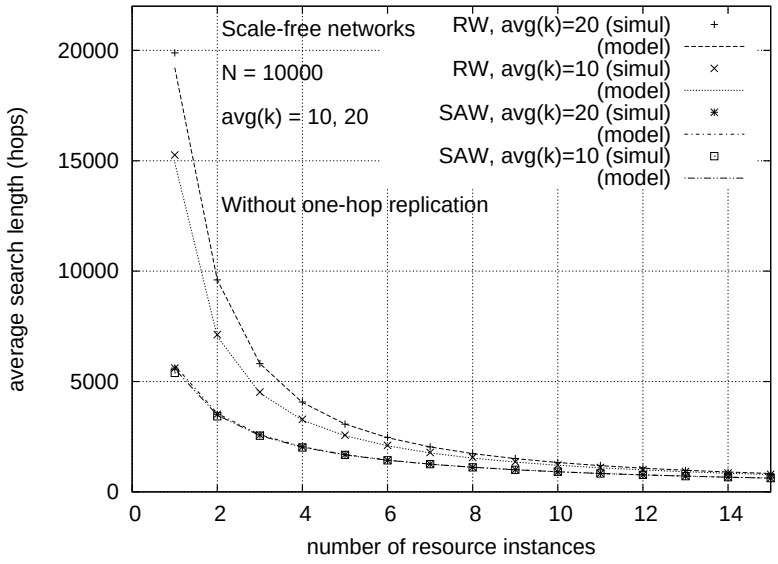


(a) Without one-hop replication.

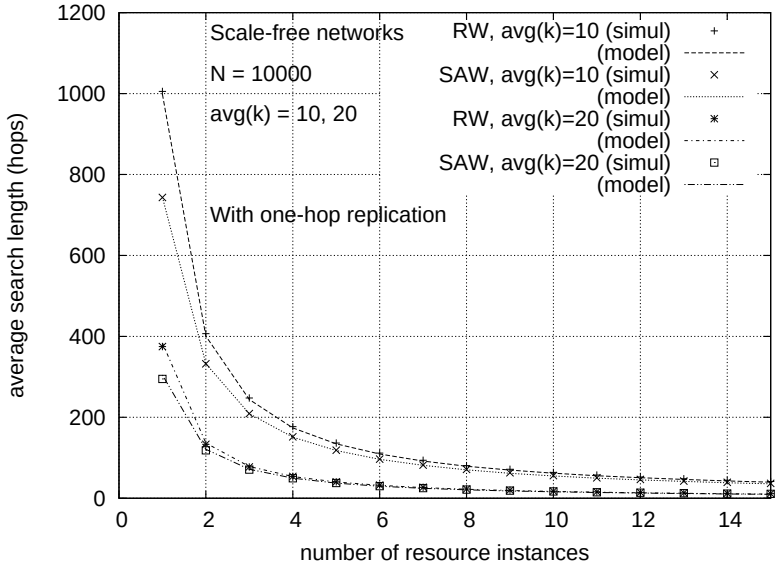


(b) With one-hop replication.

Figure 4.11: Search lengths for several resource instances in ER networks with $\overline{k} = 10, 20$.



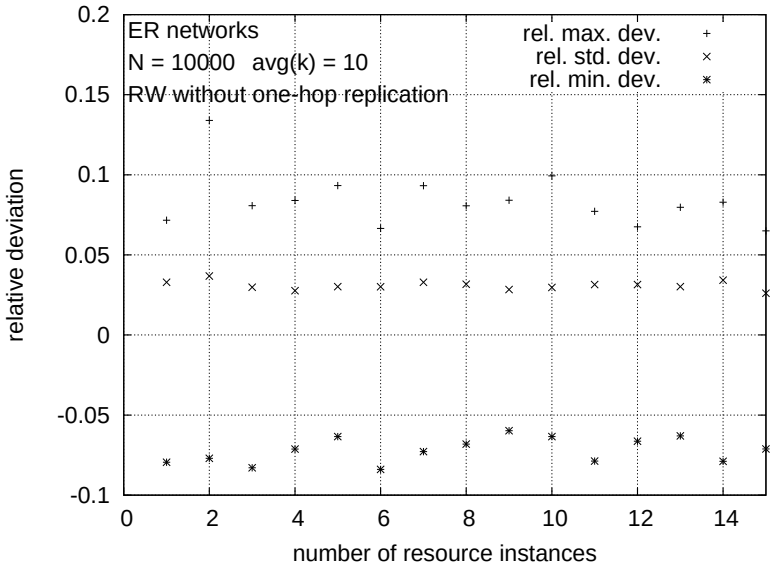
(a) Without one-hop replication.



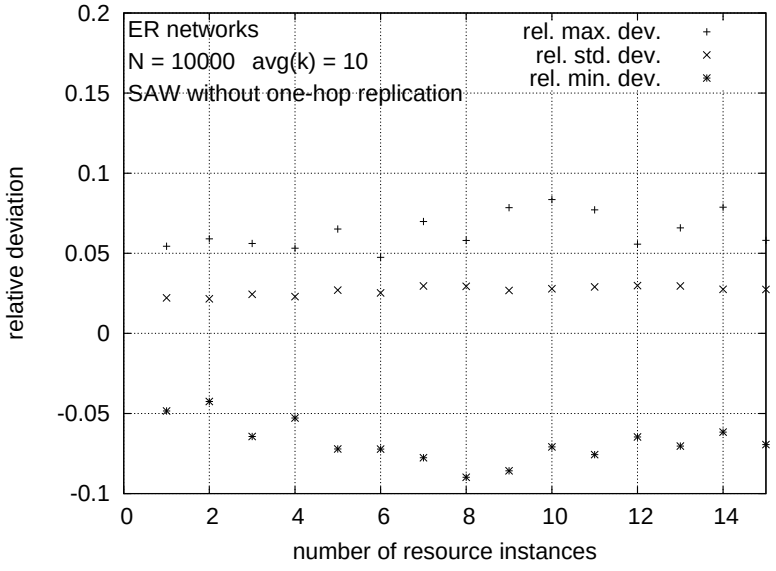
(b) With one-hop replication.

Figure 4.12: Search length for several resource instances in scale-free networks with $\overline{k} = 10, 20$.

answer this question, Figure 4.14 (for ER networks) and Figure 4.16 (for scale-free networks) show deviations of network averages with respect to the value averaged over all networks. In particular, graphs show the standard deviation and the deviations of the maximum and minimum values of network averages with respect to the total average. All deviations are given relative to the total average. Recall from the description of the simulations (Section 4.3) that *network averages* are values averaged over the 10^3 searches performed in each of the 10^2 individual networks. From these results it can be stated that the expected search length predicted by the model for a given network size and degree distribution can be taken as a reasonably good approximation of the real average search length for any (randomly built) regular, ER or scale-free network with that size and degree distribution.

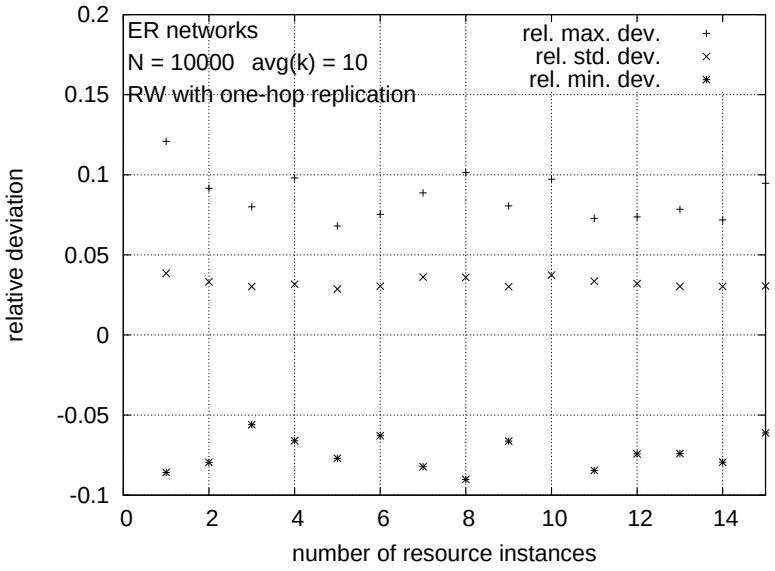


(a) RW

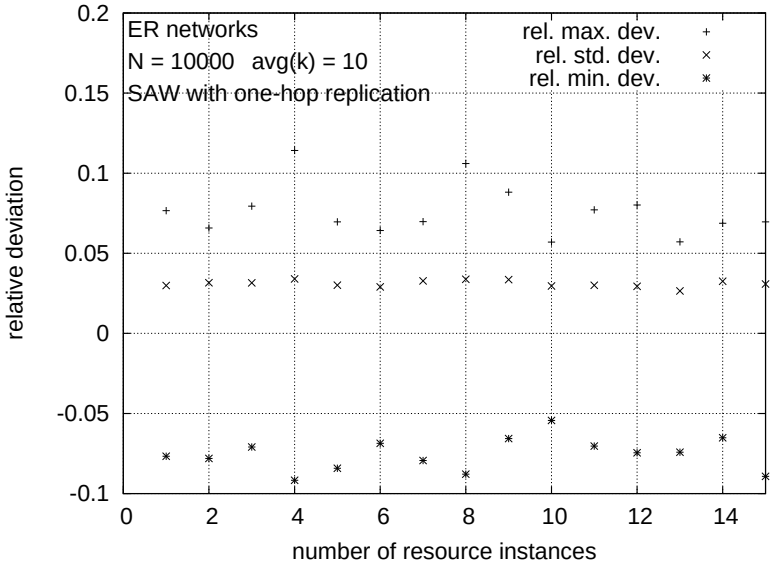


(b) SAW

Figure 4.13: Relative deviations of network averages for search lengths in ER networks with $\overline{k} = 10$ without one-hop replication.

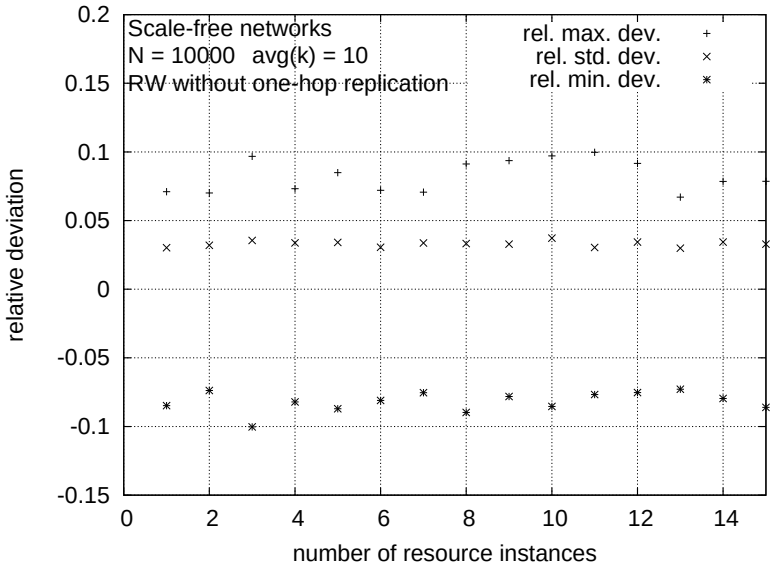


(a) RW

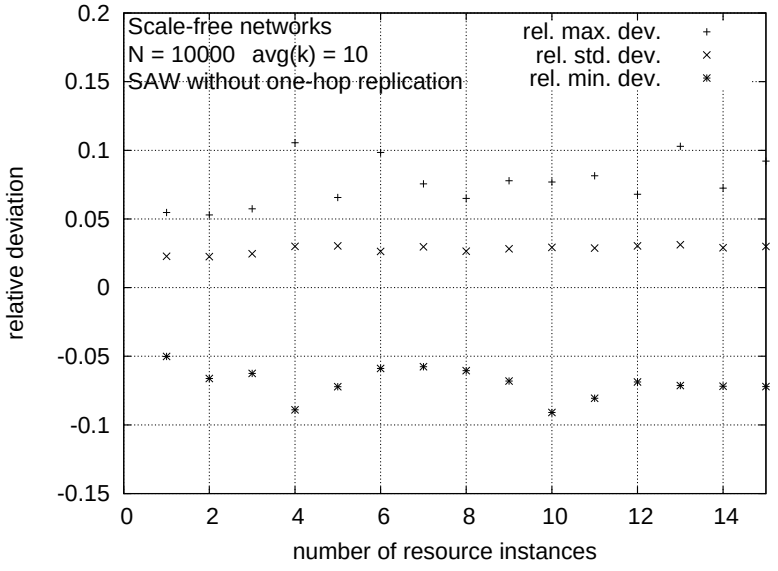


(b) SAW

Figure 4.14: Relative deviations of network averages for search lengths in ER networks with $\overline{k} = 10$ with one-hop replication.

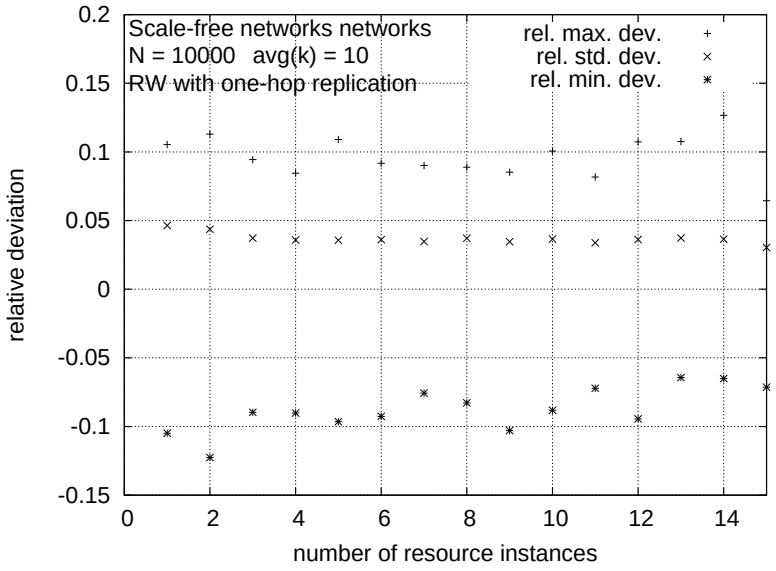


(a) RW

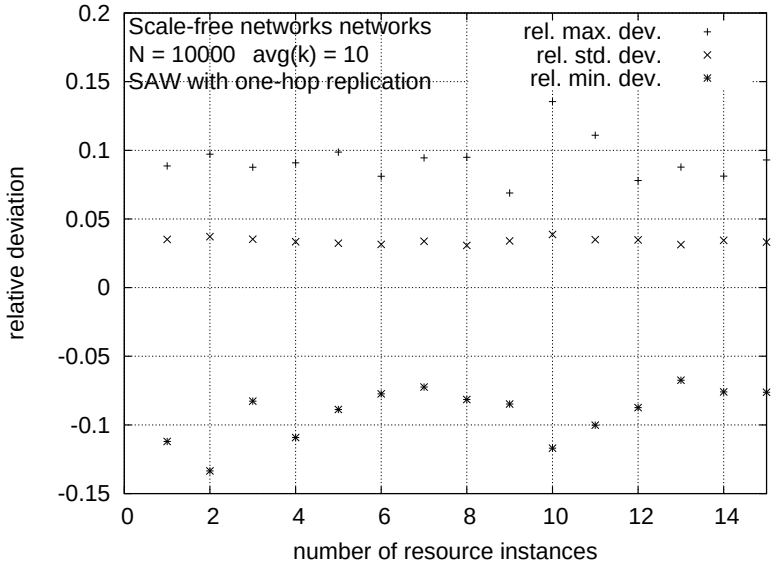


(b) SAW

Figure 4.15: Relative deviations of network averages for search lengths in scale-free networks with $\overline{k} = 10$ without one-hop replication.



(a) RW



(b) SAW

Figure 4.16: Relative deviations of network averages for search lengths in scale-free networks with $\overline{k} = 10$ with one-hop replication.

Chapter 5

Resource Location Based on Partial Random Walks

This chapter presents a family of resource location mechanisms based on the concept of *partial walks* (PW), which was introduced in Section 3.2.3. First, we explore the potential of this idea with simple RWs in the *PW-RW* mechanism. Then, we combine the concepts of PW and SAW (analyzed in Chapter 4) in the *PW-SAW* mechanism. Two variations are defined for these two mechanisms, called *choose-first* and *check-first*. We therefore present four resource location mechanisms based on PWs.

The PW mechanisms are applied to static networks in this chapter. Specifically, it is assumed that the resources that are looked remain stable in the network. Chapter 6 will apply the PW mechanisms to networks where resources exhibit a dynamic behavior, appearing and disappearing in the network nodes. A particularity of the PW mechanisms in this chapter is that they use probabilistic data structures, while mechanisms in Chapter 6 use traditional (deterministic) data structures.

A common model framework for the PW mechanisms is presented in Section 5.1. The first mechanism, choose-first PW-RW, is analyzed in Section 5.2, which also includes the evaluation of its performance. Section 5.3 presents the analysis and performance evaluation of the choose-first PW-SAW mechanism. Finally, Section 5.4 presents the check-first variation of the PW-RW and PW-SAW mechanisms.

5.1. Model

Let us consider a randomly built network of N nodes and arbitrary topology, whose nodes hold resources randomly placed in them. Resources are

unique, i.e., there is a single instance of each resource in the network. The resource location problem is defined as visiting the node that holds the resource, starting from a certain node (the *source* node). For each search, the source node is chosen uniformly at random among all nodes in the network.

The search mechanisms proposed in this work exploit the idea of efficiently building *total random walks* from *partial random walks* available at each node of the network. This process comprises two stages:

(1) Partial walks construction. Every node i in the network precomputes a set W_i of w random walks in an initial stage before the searches take place. Each of these partial walks has length s, starting at i and finishing at a node reached after s hops. In the PW-RW mechanism, the partial walks computed in this stage are simple random walks. During the computation of each partial walk in W_i, node i registers the resources held by the s first nodes in the partial walk (from i to the one before the last node). As mentioned, for generality, we assume that the resources found are stored in a Bloom filter. This information will be used in Stage 2. Bloom filters are space-efficient randomized data structures to store sets, supporting membership queries. Thus, the Bloom filter of a partial walk can be queried for a given resource. If the result is negative, the resource is not in any of the nodes of the partial walk. If the result is positive, the resource is in one of the nodes of the partial walk, unless the result was a *false positive*, which occurs with a certain probability p.[1] The size of the Bloom filters can be designed for a target (small) p considered appropriate. A variation of the partial walk construction mechanism consists of using PWs that are *self-avoiding* walks (SAW). The resulting mechanism, called PW-SAW, is analyzed in Section 5.3.

(2) The searches. After the PWs are constructed, searches are performed in the following fashion when the choose-first PW-RW/PW-SAW mechanisms are used. When a search starts at a node A, a PW in W_A is chosen uniformly at random. Its Bloom filter is then queried for the desired resource.

- If the result is negative, the search *jumps* to node B, the last node of that partial walk. Note that the current node and the node to which the search *jumps* are not neighbors in the overlay network in general. Jumps therefore make use of the underlying network[2].

[1] More concretely, p is the probability of obtaining a positive result conditioned on the desired resource not being in the filter.

[2] In fact, neighbors in the overlay network are not neighbors in general in the underlying network either (see Section 3.3.1). Therefore, both jumps and normal steps make use of the underlying network.

The process is then repeated at B, so that the search keeps jumping in this way while the results of the queries are negative.

- If, when at a node C, the query to the Bloom filter (of the PW randomly chosen from W_C) gives a positive result, the search *traverses* that partial walk looking for the resource until the resource is found or the partial walk is finished.

  If the resource is found, the search stops. If the search reaches the last node D of the partial walk without having found the resource in the previous nodes, it means that the result of the Bloom filter query was a false positive. The search then randomly chooses a partial walk in W_D and decides whether to jump over it or to traverse it depending on the result of the query to its Bloom filter, as described above.

A variation of this behavior consists of first checking all PWs of the node for the desired resource, and then randomly choosing among the ones with a positive result. The resulting mechanisms, called check-first PW-RW/PW-SAW are analyzed in Section 5.4.

In this work, we are interested in the number of *hops* to find a resource (when PWs of length s are used), which is defined as the *search length* and denoted L_s. Some of these hops are *jumps* (over PWs) and other are *steps* (traversing PWs). In turn, we distinguish between *trailing steps*, if they are the ones taken when the resource is found, and *unnecessary steps*, if they are taken when the resource is not found. The search length is a random variable that takes different values when independent searches are performed. The *search length distribution* is defined as the probability distribution of the search length random variable. We are interested in finding the *expected search length*, denoted $\overline{L}_s$. Figure 5.1 summarizes the behavior of the search mechanisms.

At this point, we emphasize the difference between the *search* just defined and the *total walk* that supports it, consisting of the concatenation of *partial walks* as defined above. Searches are shorter in length than their corresponding total walks because of the number of steps saved in jumps over partial walks in which we know that the resource is not located (although these saving may be reduced by the unnecessary steps due to Bloom filter false positives).

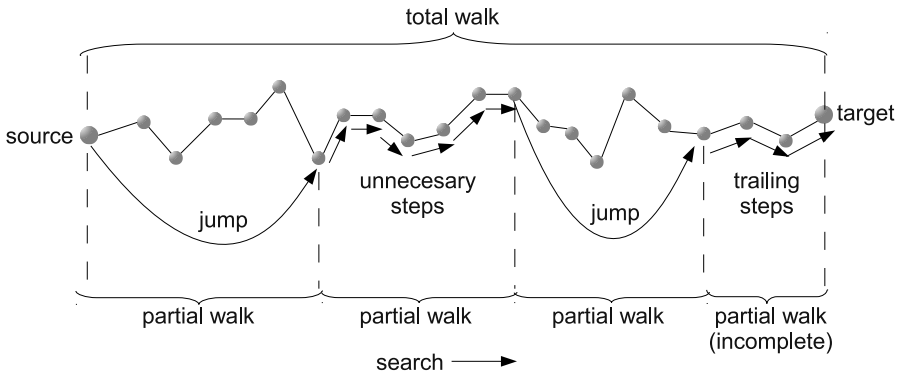


Figure 5.1: An example of search, using PWs of length $s = 6$.

5.2. Choose-First PW-RW

This section presents the analysis and performance evaluation of the choose-first PW-RW resource location mechanism.

5.2.1. Analysis

We make an additional assumption in order to simplify this analysis. Once a PW has been used in the total walk of a search, it is never reused again in that total walk or in any other searches. Thus we guarantee that the total walks are true random walks. This implies that in practice each node needs to have a large number of precomputed partial walks (w), assumption that would compromise the benefits of the proposed mechanism in practice. Simulations in Section 5.2.2 show that real cases with small w behave very similarly to the base case provided by this analysis.

Let L_s be the random variable representing the number of hops in the search (i.e., its length) when PWs of length s are used. The expected search length is denoted by $\overline{L}_s$. Let L be the random variable representing the number of hops of the corresponding total walk. Its expected search length is denoted $\overline{L}$. Making use of the assumption that partial walks are never reused, L can be viewed as the length of a search based on a simple random walk in the considered network, and $\overline{L}$ as the expected search length of random walks in that network (that is, the $\overline{L}_{rw}$ defined in Chapter 4). Then, we can state the following theorem:

Theorem 1 *If the expected number of trailing steps is assumed to be uniformly distributed in* $[0, s-1]$[3]*, then the expected search length is:*

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) \cdot (1-p) + \overline{L} \cdot p. \tag{5.1}$$

Proof. Let P, J, U and T be random variables representing the number of partial walks, jumps, unnecessary steps and trailing steps in a search, respectively. Their expectations are denoted as $\overline{P}$, $\overline{J}$, $\overline{U}$ and $\overline{T}$. Since hops in a search can be jumps, unnecessary steps or trailing steps, it follows that, $L_s = J + U + T$. Then, the expected search length for partial walks of size s is[4] $\overline{L}_s = \overline{J} + \overline{U} + \overline{T}$.

The expected number of jumps can be obtained from the expected number of partial walks in the search ($\overline{P}$) and from the probability of false positive (p) as $\overline{J} = \overline{P} \cdot (1-p)$, since J follows a binomial distribution $B(P, 1-p)$, where the number of experiments is the random variable representing the number of partial walks in a search (P) and the success probability is the probability of obtaining a negative result in a Bloom filter query $(1-p)$.[5]

For the expected number of unnecessary steps, $\overline{U} = \overline{P} \cdot p \cdot s$, since $\overline{P} \cdot p$ is the expected number of false positives in the search and each of them contributes with s unnecessary steps. The number of partial walks in a search can be obtained dividing the length of the total walk by the size of a partial walk: $P = \left\lfloor \frac{L}{s} \right\rfloor = \frac{L-T}{s}$. Then, the expected number of partial walks in a search is $\overline{P} = \frac{\overline{L}-\overline{T}}{s}$.

Since we assume that the expected number of trailing steps is uniformly distributed between 0 and $(s-1)$, its expectation is $\overline{T} = \frac{s-1}{2}$.

Using the previous equations we have:

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) + p \cdot \left(\overline{L} - \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right)\right),$$

[3]This is, in fact, a pessimistic assumption. The distribution of trailing steps is approximately uniform, but shorter walks have a slightly higher probability than longer ones. This can be shown analytically and has been confirmed in our experiments (see paragraph *Distribution of the Number of Trailing Steps* later in this section). Therefore, the expected value in our analysis, derived from a perfectly uniform distribution, is slightly higher than the real average value.

[4]In the following, we make implicit use of the linearity properties of expectations of random variables.

[5]If Y is a random variable with a binomial distribution with success probability p, in which the number of experiments is in turn the random variable X, it can be easily shown that $\overline{Y} = \overline{X} \cdot p$ (see paragraph *Expectation of a Random Variable with a Binomial Distribution in which the Number of Experiments is Another Random Variable* later in this section).

where the first term is the expectation of the search length for a "perfect" Bloom filter (one that never returns a false positive when the resource is not in the filter, i.e., $p = 0$), and the second term is the expectation of the additional search length due to false positives ($p \neq 0$).

Another interpretation of this expression is obtained if we reorganize it to make explicit the contributions of a perfect filter and of a "broken" filter (one that always returns a false positive result when the resource is not in the filter, i.e., $p = 1$) as

$$\overline{L}_s = \left(\frac{s}{2} + \frac{2\overline{L}+1}{2s} - 1\right) \cdot (1-p) + \overline{L} \cdot p.$$

□

From this theorem and using calculus, we have the following corollary.

Corollary 2 *The optimal length of the partial walks, i.e., the length of the partial walks that minimizes the expected search length, is:*

$$s_{opt} = \sqrt{2\overline{L}+1}.$$

The obtained value needs to be rounded to an integer, which is omitted in the notation. Observe that *the optimal length of the partial walks is independent from the probability of false positives in the Bloom filters*, while the expected search length ($\overline{L}_s$) does of course depend on it.

Corollary 3 *The optimal expected search length, i.e., the expected search length when partial walks of optimal length are used, is:*

$$\overline{L}_{opt} = \left(\sqrt{2\overline{L}+1} - 1\right)(1-p) + \overline{L}\,p = (s_{opt}-1)(1-p) + \overline{L}\,p. \qquad (5.2)$$

This result is an interesting relation between the optimal length of the search and the optimal length of the PWs. If we consider perfect Bloom filters ($p = 0$), we have $\overline{L}_{opt} = s_{opt} - 1$, which for large $\overline{L}$ (e.g. for large networks) becomes $\overline{L}_{opt} \approx s_{opt}$. Therefore, we have found that, for large N and $p = 0$, *the optimal expected search length approximately equals the optimal length of the partial walks.* For arbitrary values of p, Equation 5.2 shows that $\overline{L}_{opt}$ is linear in p.

This completes the analysis of choose-first PW-RW. Section 5.3.3 provides an alternative analysis using a different approach. Instead of assuming that the total walk is a random walk, it considers that it is built using the w PWs available at each node, which avoids the need of $\overline{L}$. On the other hand, the alternative model does not provide expressions for the optimal PW length or the expected search length.

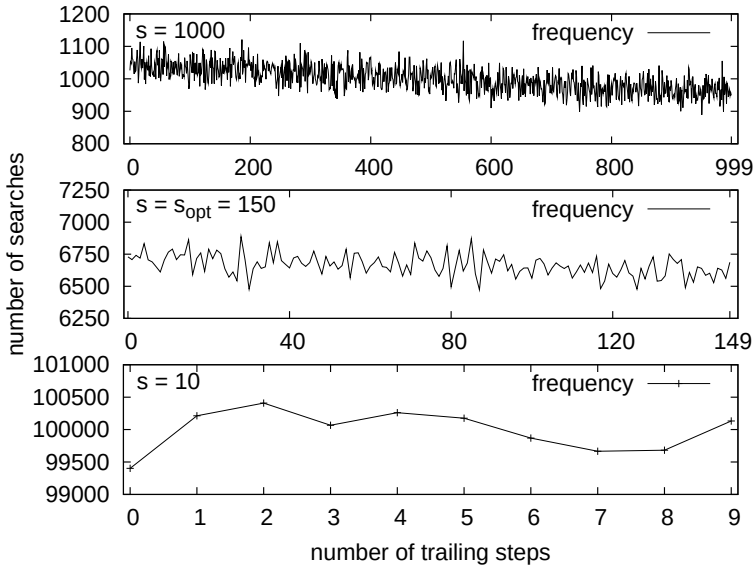


Figure 5.2: Distributions of the number of trailing steps in the regular network.

Distributions of the Number of Trailing Steps

The previous proof of Theorem 1 assumes that the distribution of the number of trailing steps in the last partial walk until the search finds the resource is uniform between 0 and $s-1$, corresponding to the cases where the first node/last node in the partial walk holds the desired resource. Recall that the Bloom filter stores the resources held by the s first nodes in the partial walk, from the node that precomputed the partial walk to the one before its last node (which is included in the partial walks departing from it). We have obtained that distribution from the 10^6 searches in our experiment for each of the three networks. Figure 5.2 shows the distributions for the regular network when $s = 10$, $s = s_{opt} = 150$ and $s = 1000$. Distributions for the ER and scale-free networks are similar in shape.

It is observed that there is a slight decrease on the frequency as the number of steps grows. This is due to the fact that the number of trailing steps is essentially the length of the total walk modulus the length of partial walks (s). The total walk is a random walk, and its distribution can be obtained approximately by Equation 5.3 below.[6] Since it is a decreasing function, as it is shown below, the frequency on the left end of an interval of width s is always higher than the frequency on the right end, thus accounting

[6]The distribution of simple random walk searches has also been obtained experimentally, showing that Equation 5.3 is a good approximation.

for the observed decrease.

This means that the result provided by Theorem 1 is *pessimistic*, since the estimated average number of trailing steps is slightly higher than the real one. Results in Section 5.2.2 have shown that values of average search lengths predicted by Equation 5.1 are very similar to values computed from simulations, with larger error for higher values of s.

The probability distribution of simple random walk searches can be estimated using Equation 5.3. It can be demonstrated that it is strictly decreasing, that is: $P_i - P_{i-1} < 0$ for $0 \leq i < \infty$, as follows:

$$\begin{aligned} P_0 &= \frac{1}{N}, \\ P_i &= \left(1 - \sum_{j=0}^{i-1} P_j\right) \cdot \frac{1}{N-1}, \text{ for } i > 0. \end{aligned} \tag{5.3}$$

First, it is shown by induction that $0 < \sum_{i=0}^{k} P_i < 1$ for $k \geq 0$ and $N > 0$. It holds trivially for $k = 0$. Then, it is also true for $k > 0$ if it holds for $k-1$:

$$\begin{aligned} \sum_{i=0}^{k} P_i &= \sum_{i=0}^{k-1} P_i + \left(1 - \sum_{i=0}^{k-1} P_i\right) \cdot \frac{1}{N-1} \\ &= \frac{N-2}{N-1} \cdot \sum_{i=0}^{k-1} P_i + \frac{1}{N-1} \\ &< \frac{N-2}{N-1} + \frac{1}{N-1} = 1. \end{aligned}$$

Next, it is shown that $0 < P_i < 1$ for $i \geq 0$ as a corollary of the previous result. It is checked for $i = 0$ by inspection. For $i > 0$, we have that $P_i = \left(1 - \sum_{j=0}^{i-1} P_j\right) \cdot \frac{1}{N-1}$. By the previous result:

$$0 < 1 - \sum_{j=0}^{i-1} P_j < 1,$$

then we have that:

$$0 < P_i = \left(1 - \sum_{j=0}^{i-1} P_j\right) \cdot \frac{1}{N-1} < 1.$$

Finally, it is shown that $P_i - P_{i-1} < 0$ for $i > 0$. For $i = 1$, it is checked

by inspection. For $i > 1$:

$$\begin{aligned} P_i - P_{i-1} &= \left(1 - \sum_{j=0}^{i-1} P_j\right) \frac{1}{N-1} - \left(1 - \sum_{j=0}^{i-2} P_j\right) \frac{1}{N-1} \\ &= -\frac{P_{i-1}}{N-1}. \end{aligned}$$

Since we have shown that $0 < P_{i-1} < 1$, it follows that $P_i - P_{i-1} < 0$.

Expectation of a Random Variable with a Binomial Distribution in which the Number of Experiments is Another Random Variable

Let X be a random variable with sample space $S = \mathbb{N}_0 = \{0, 1, 2 \ldots\}$. Let Y be a random variable representing the number of successes when X experiments are performed with a success probability p. Y has a binomial probability distribution $Y \sim \mathrm{B}(X, p)$, where the number of experiments is, in turn, a random variable. Then, from the definition of expectation and applying the Total Probability Theorem, the expectation of Y is $\mathrm{E}[Y] = \mathrm{E}[X] \cdot p$.

$$\begin{aligned} \mathrm{E}[Y] &= \sum_{y=0}^{\infty} y \cdot \mathrm{P_r}[Y = y] \\ &= \sum_{y=0}^{\infty} y \cdot \left\{ \sum_{x=0}^{\infty} \mathrm{P_r}[Y = y | X = x] \cdot \mathrm{P_r}[X = x] \right\} \\ &= \sum_{x=0}^{\infty} \mathrm{E}[Y | X = x] \cdot \mathrm{P_r}[X = x] \\ &= \sum_{x=0}^{\infty} x \cdot p \cdot \mathrm{P_r}[X = x] = \mathrm{E}[X] \cdot p. \end{aligned}$$

Cost of Precomputing PWs

Since searches use the partial walks precomputed by each of the nodes of the network, the cost of this computation must be taken into account. We measure this cost as the number of messages C_p that need to be sent to compute all the PWs in the network. This quantity has been chosen to be consistent with our measure of the performance of the searches. Indeed, each *hop* taken by a search can be alternatively considered as a *message* sent. In addition, C_p is independent from other factors like the processing power of nodes, the bandwidth of links and the load of the network. The cost of

precomputing a set of PWs can be simply obtained as $C_p = Nw(s+1)$, since each of the N nodes in the network computes w partial walks, sending s messages to build each of them plus one extra message to get back to its source node.

Let's suppose that each node starts on the average b searches that are processed by the network with the set of PWs precomputed initially. We define C_s to be the total number of messages needed to complete those searches. If the expected number of messages of a search is $\overline{L}_s + 1$ (counting the message to get back to the source node), we have that $C_s = Nb(\overline{L}_s + 1)$. Now, defining C_t as the *average total cost per search*, we can write:

$$C_t = \frac{C_s + C_p}{Nb} = (\overline{L}_s + 1) + \frac{w}{b}(s+1). \tag{5.4}$$

The second term in Equation 5.4 is the contribution to the cost of the precomputation of the PWs. This contribution will remain small provided that the number of searches per node in the interval is large enough.

5.2.2. Performance Evaluation

The goal of this section is to apply the model for choose-first PW-RW presented in the previous section to real networks, and to validate its predictions with data obtained from simulations. The experimental framework described in Section 3.5 is used. Three types of networks have been chosen for the experiments: regular networks (constant node degree), Erdős-Rényi (ER) networks and scale-free networks (see Section 3.3.4). A network of each type and size $N = 10^4$ has been randomly built with the method proposed by Newman et al. [67] for networks with arbitrary degree distribution, setting their average node degree to $\overline{k} = 10$. For each experiment, 10^6 searches have been performed, with the source node chosen uniformly at random among the N nodes. Likewise, the resource has been placed in a node chosen uniformly at random for each experiment.

Optimal PW Size and Expected Search Length

We start by applying Theorem 1 to the networks described above to obtain the expected search length as a function of the size of the PWs.[7]

[7]For each network, the expected length of a random walk search ($\overline{L}$) is needed. We estimate these expected values by simulating 10^6 simple random walk searches and averaging their lengths in each of the networks (these average search lengths are denoted using lowercase ($\bar{l}$) to distinguish them from the actual expected value ($\overline{L}$) in the model. The values obtained from the experiments are: $\bar{l}_{reg} = 11246$, $\bar{l}_{ER} = 12338$, and $\bar{l}_{sf} = 15166$).

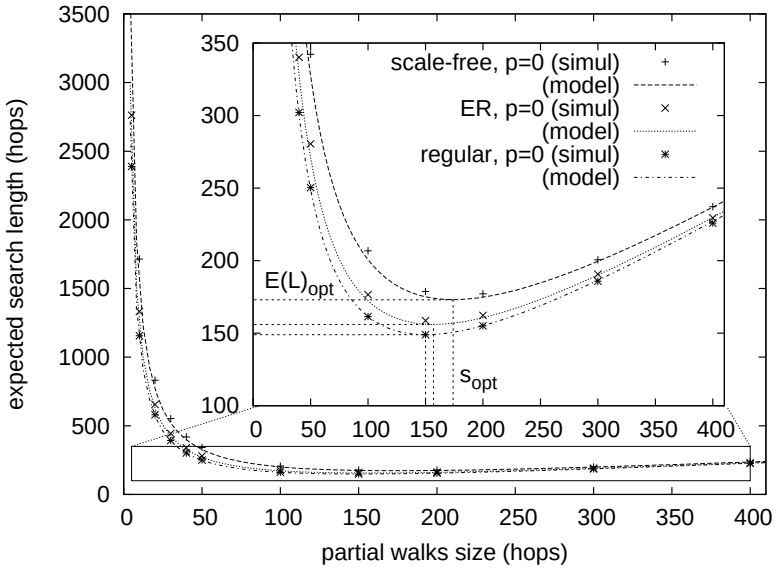


Figure 5.3: Expected search length ($\overline{L}_s$) as a function of s when $p = 0$ in a regular network, an ER network and a scale-free network. The optimal points $(s_{opt}, \overline{L}_{opt})$ for each network are $(150, 149)$, $(157, 156)$, and $(174, 173)$.

Figure 5.3 provides plots of the expected search lengths ($\overline{L}_s$) given by Equation 5.1 as a function of the size of the PWs (s), when the probability of a false positive in the Bloom filter is set to $p = 0$, for the three types of networks considered. Results from the analytical model are shown as curves while simulation data are shown as points. The curves for the three networks show a minimum point $(s_{opt}, \overline{L}_{opt})$. This behavior is due to the fact that, when s is small, the number of jumps needed to reach a PW containing the chosen resource grows, therefore increasing the value of $\overline{L}$. In turn, for larger values of s, the number of trailing steps within the last PW grows, also increasing the value of $\overline{L}$.

Figure 5.4 illustrates (using Equation 5.2 and taking into account the fact that s_{opt} is independent from the value of p) the optimal expected search length ($\overline{L}_{opt}$) as a function of the probability of false positives (p). It can be seen that it grows linearly: the regular network exhibits the smallest slope, followed by the ER network and then by the scale-free network. For $p = 1$, Equation 5.2 degenerates to $\overline{L}_{opt} = \overline{L}$, since the search performs all the hops of the total walk (i.e., it is a random walk). In fact, Equation 5.1

These results agree with the analytical method presented in Chapter 4 (a modification of the one provided in [78]), which produces the following results: $\bar{l}_{reg} = 11095$, $\bar{l}_{ER} = 12191$, and $\bar{l}_{sf} = 14920$.

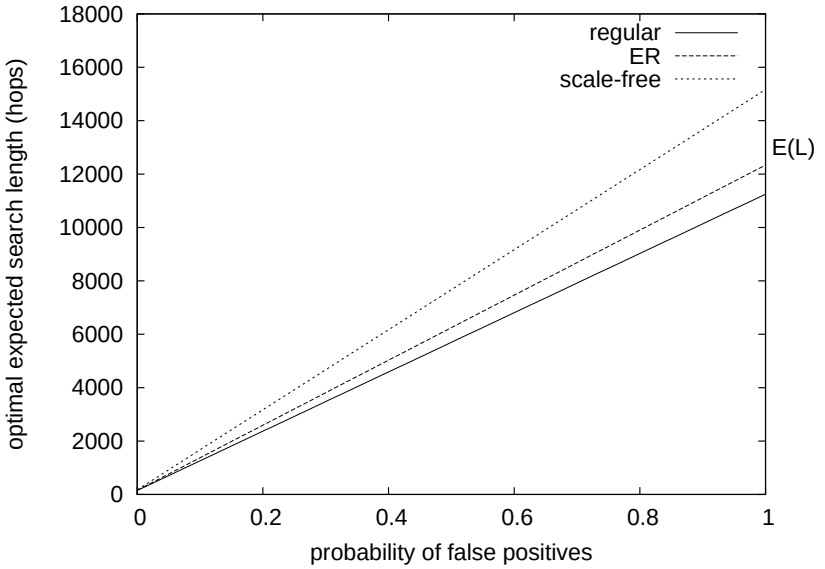


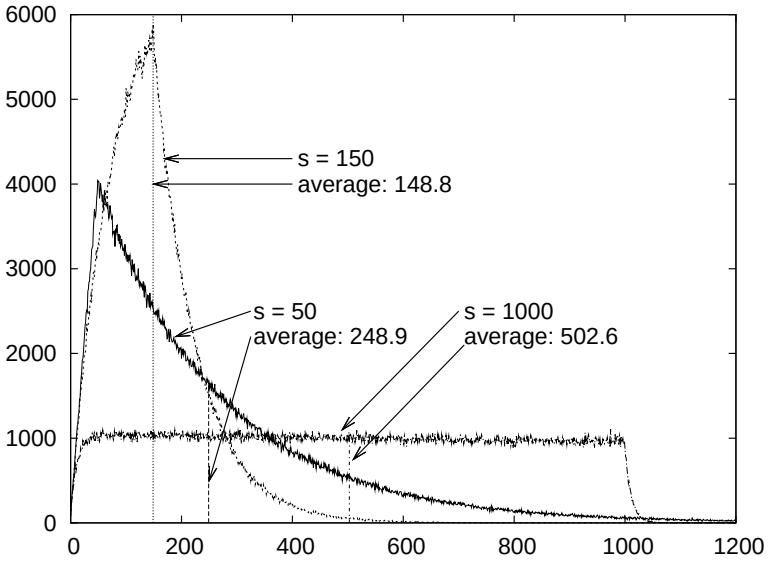
Figure 5.4: Optimal expected search length ($\overline{L}_{opt}$) as a function of p.

also degenerates to $\overline{L}_s = \overline{L}$ in this case, meaning that the expected search length is that of random walk searches regardless the size of the PWs (s).

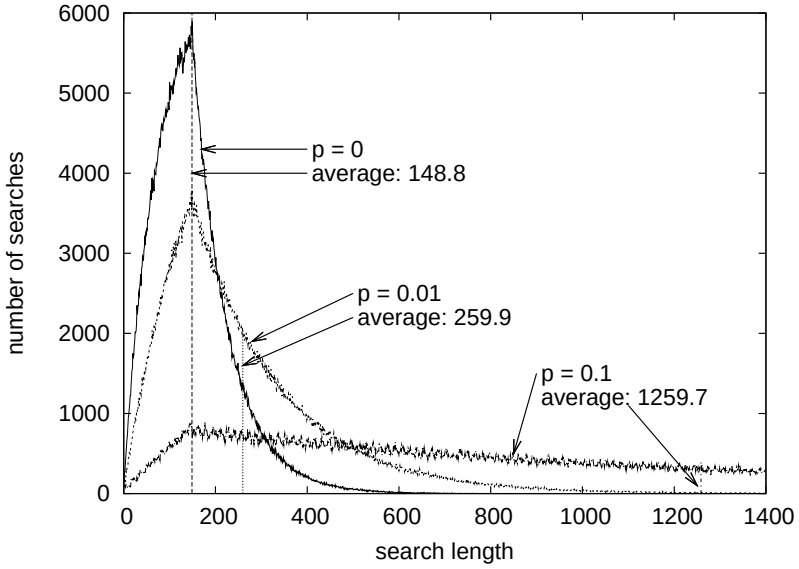
Length Distributions

The aim of this section is to experimentally explore how the use of PWs affects the statistical distribution of search lengths. We first obtain the lengths distributions of searches using PWs that are never reused. Later in this section we will discuss the effect of having a limited number of partial random walks that are reused. We consider each random walk to be the total walk of a search based on PWs. For each original random walk, we break it in pieces of size s, which are taken as the PWs that make up the total walk. Then we consider a search that uses those PWs and count the number of hops (jumps plus trailing steps plus unnecessary steps). This gives the length of the search if it had been constructed using those (precomputed) PWs. Note that the PWs are not reused because they are obtained from independent (real) random walks.

The search length distributions in the regular network for $p = 0$ and for several values of s are shown in Figure 5.5(a). The plots also show, as vertical bars, the average search lengths computed from each distribution. These average values are very close to the expected values calculated with Equation 5.1 ($\overline{L}_{50} = 248.9$, $\overline{L}_{150} = 149.0$ and $\overline{L}_{1000} = 510.2$). Therefore, our model accurately predicts average lengths of searches based on PWs of size



(a) Search lengths for $p = 0$ and for $s = s_{opt} = 150$, $s = 50$ and $s = 1000$.



(b) Search lengths for s_{opt} and for $p = 0, 0.01, 0.1$.

Figure 5.5: Distributions of search lengths (histograms) with PWs that are not reused in the regular network.

s in the three types of networks considered in our experiments.

As for the shape of the distributions, we observe that for low s ($s = 50$ in Figure 5.5(a)) the search lengths are dominated by the number of jumps, which is proportional to the length of the total walk. On the other hand, for high s ($s = 1000$ in Figure 5.5(a)) the distribution adopts a rather uniform shape. Search lengths are dominated here by the number of trailing steps in the last PW, and this has approximately an uniform distribution between 0 and $s - 1$, as mentioned earlier. The optimal length for the PWs, s_{opt} ($s = 150$ in Figure 5.5(a)), represents a transition point between these two effects. The shape is such that the values around the average search length (which approximately equals s_{opt}, according to Equation 5.2) are also the most frequent.

Once it has been found the optimal length for the PWs s_{opt} (which is known to be independent of the value of p), we investigate the effect of the probability of false positive of Bloom filters in these distributions. Figure 5.5(b) shows the distributions of search lengths (histograms) for the regular network when $s = s_{opt}$ and for several values of p. It can be seen that the distributions get wider and lower as p grows, pushing average search lengths to higher values, in accordance with Figure 5.4. However, we observe that the most frequent lengths remain the same regardless of the value of p. For $p = 0$, the most frequent value for each network approximately equals the average search length which, in turn, approximately equals the optimal length of the PWs ($s_{opt} = 150$ for the regular network). For greater values of p, the average search length grows while the most frequent value stays the same.

Regarding the distributions for the ER and the scale-free networks, they have similar shapes and are not shown here. However, we have used these distributions to obtain Table 5.1 (explained below).

Effect of Reusing PWs

At this point, we note that we have been assuming that PWs are never reused. In this section, we explore the distributions when the total walks are built reusing a limited number w of partial walks precomputed in each node. This is in contrast with our initial assumption that precomputed partial walks are not reused in searches. Here, we attempt to answer the question "How many partial walks does a node need to precomputed, for the search lengths distribution to be similar to that corresponding to never reusing partial walks?". Our results show that, for the networks considered in our experiment, and for the optimal partial walk length (s_{opt}), it is enough to have as few as *two* precomputed partial walks in every node. The extreme

case of having just *one* precomputed partial walk yields a significant fraction of unfinished searches, since it is relatively easy to build walks that are loops that do not visit all the nodes. Indeed, if the last node of a partial walk is a node whose (only) partial walk has been previously used in that total walk, it will take the search to the same place again, resulting in a never-ending loop. However, if a node has several partial walks, and the search chooses one randomly among them (for the next jump or partial walk traversal), the chances of entering a loop are very small.

Figures 5.6 to 5.8 show the search lengths distributions in the regular network. The top plots of these figures show the length distributions of searches based on PWs that are not reused. The middle and bottom plots show the length distributions of searches based on reusing a single partial walk or two partial walks per node, respectively.

We note that the shape of the distributions is the same for all values of w. However, distributions for $w = 1$ are lower, and the average search length (marked as a vertical bar) is also smaller. This is due to a significant percentage of unfinished searches (about 26%), left out of the histograms, due to loops as explained above. If we focus now on the distributions for $w = 2$, we observe that both the distribution and the average search length are very similar to those for PWs that are not reused. We have performed additional experiments with higher values of w, confirming this observation. This suggests that just two precomputed partial walks per node are enough to obtain a behavior close to the theoretical case of using PWs that are not reused. The distributions of searches in the ER network and the scale-free network are omitted here, since their shape and the conclusions drawn are the same as for the regular network.

We now measure the difference between the search length distributions for several values of w and the base case of not reused PWs. In Figures 5.9 to 5.11 we plot these (signed) differences for $w = 2$ and several values of p in the regular network. It is observed that differences are small for low values of p, growing as p gets bigger. But the magnitude of the differences seem to be within the order of variation of the values of the histograms for all values of p. As a global measure of the difference between the distributions for $w = 2$ and for PWs that are not reused we compute the *mean relative difference* as $\frac{1}{L_{0.9}+1} \sum_{l=0}^{L_{0.9}} \frac{|h_2(l)-h_{\text{af}}(l)|}{h_{\text{af}}(l)}$, where $h_w(l)$ is the number of searches with length ℓ when using w partial walks per node, and $h_{\text{af}}(l)$ corresponds to the case of not reused PWs. The tail of long searches with low frequency is removed from the calculation, since those values yield high relative differences that distort the measurement. For this, the summation includes 90% of the searches, from length zero up to $L_{90\%}$, where $L_{90\%}$ is the 90% percentile of search

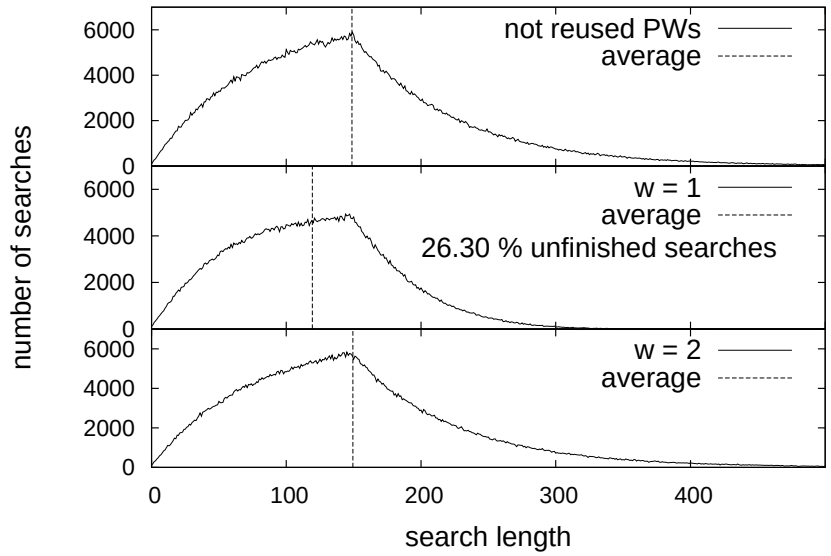


Figure 5.6: Search length distributions for PWs that are not reused, for $w = 1$ and for $w = 2$, in the regular network ($p = 0$).

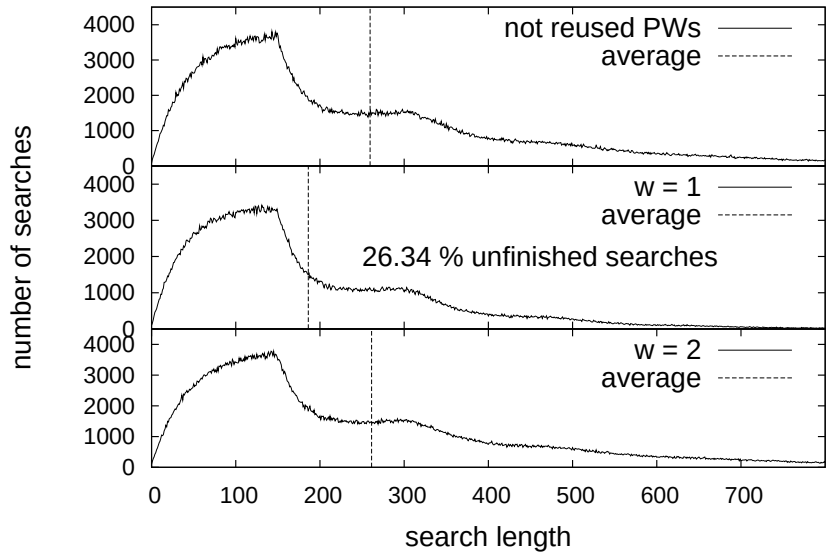


Figure 5.7: Search length distributions for PWs that are not reused, for $w = 1$ and for $w = 2$, in the regular network ($p = 0.01$).

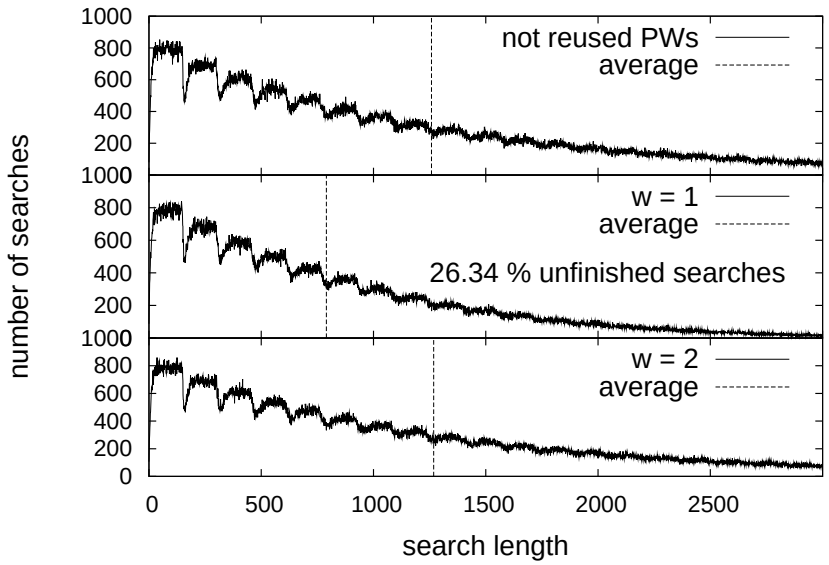


Figure 5.8: Search length distributions for PWs that are not reused, for $w = 1$ and for $w = 2$, in the regular network ($p = 0.1$).

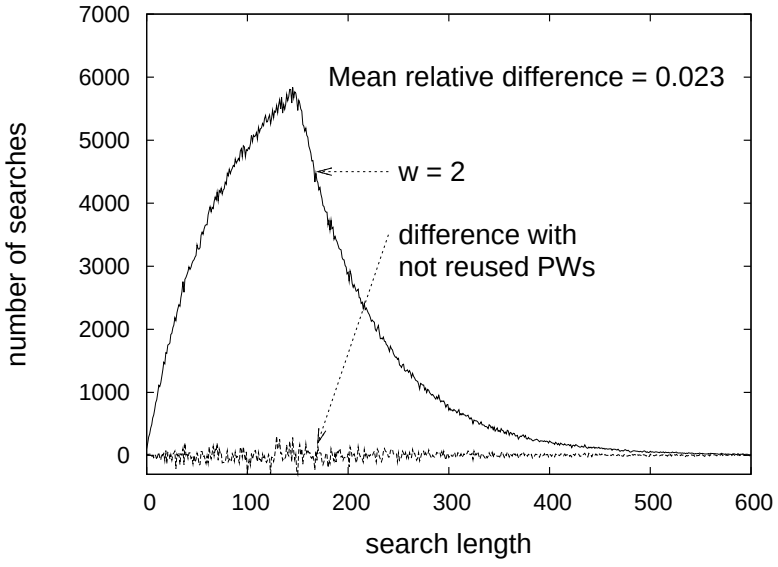


Figure 5.9: Difference between search length distributions for $w = 2$ and for non-reused PWs in the regular network ($p = 0$).

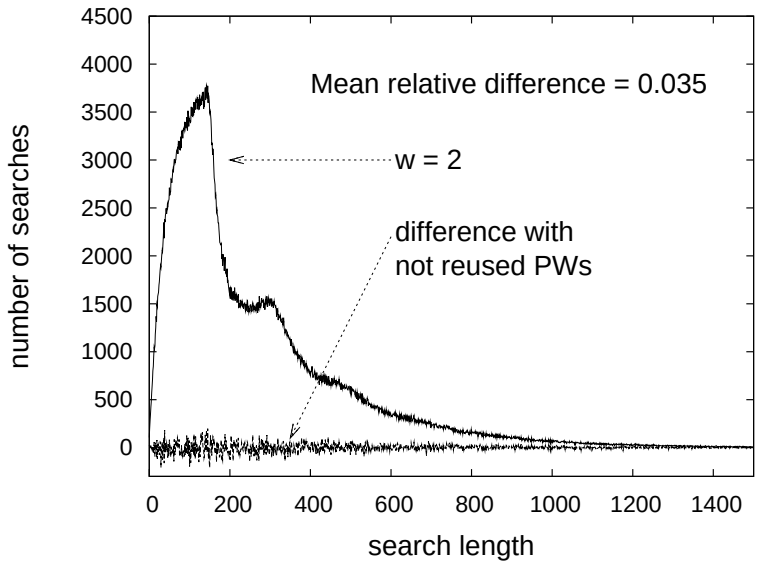


Figure 5.10: Difference between search length distributions for $w = 2$ and for non-reused PWs in the regular network ($p = 0.01$).

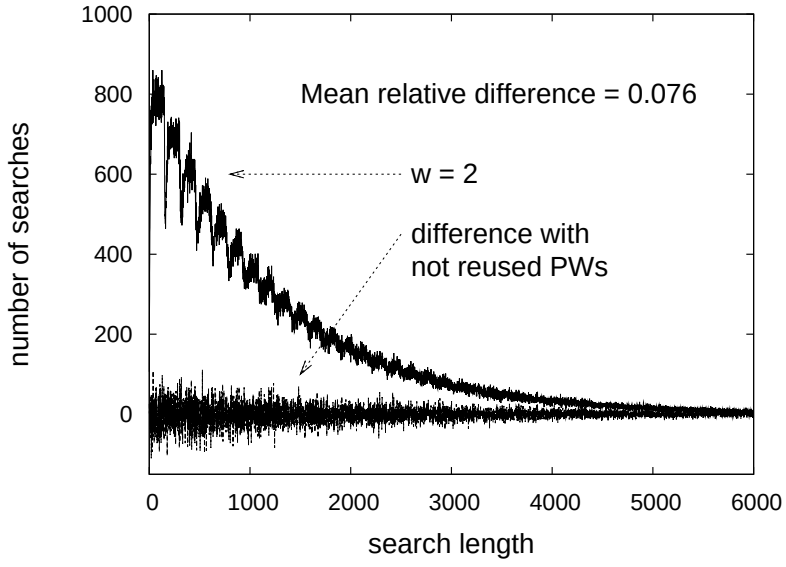


Figure 5.11: Difference between search length distributions for $w = 2$ and for non-reused PWs in the regular network ($p = 0.1$).

	Reduction of $\bar{l}$ (%)		
Network type	$p = 0$	$p = 0.01$	$p = 0.1$
Regular	98.67	97.68	88.73
ER	98.71	97.68	88.42
Scale-free	98.83	97.79	88.43

Table 5.1: Reductions of average search lengths of choose-first PW-RW with respect to RW.

lengths. The mean relative differences for $p = 0$, $p = 0.01$ and $p = 0.1$ are, respectively, 0.023, 0.035 and 0.076.

Therefore we conclude that, for the types of networks in our experiment, just two precomputed partial walks per node are enough to obtain searches whose lengths are statistically similar to those that would be obtained with PWs that are not reused.

Comparison of Performance with Respect to RW Searches

Finally, in Table 5.1 we compare the performance of the proposed search mechanism with respect to random walk searches. We can see that the reduction in the average search length that PW-RW achieves with respect to simple random walk is lower for higher p, ranging from around 98% in the case when $p = 0$ to 88% when $p = 0.1$. Furthermore, we also see that the achieved reductions are independent of the network type.

5.3. Choose-First PW-SAW

As it was pointed in Section 5.1 when we introduced the PW construction mechanism in Stage 1, a possible variation consists of using self-avoiding walks (SAW) instead of simple random walks. The resulting search mechanism is called PW-SAW. The basic idea is to revisit less nodes, thus increasing the chances of locating the desired resource. In short, a SAW chooses the next node to visit uniformly at random among the neighbors that have not been visited so far by the walk. If all neighbors have already been visited, it chooses uniformly at random among all neighbors, like a simple random walk.

5.3.1. Analysis

When PWs are self-avoiding walks, their concatenation is not a random walk, and hence Theorem 1 is no longer valid. We state a new theorem for the choose-first PW-SAW mechanism, proving it using a different approach.

Theorem 4 *If the expected number of trailing steps is assumed to be uniformly distributed in* $[0, s-1]$*, then the expected search length of PW-SAW is*

$$\overline{L}_s = \frac{1}{N} \sum_k n_k \left(\frac{1}{P_{tp}(k)} \cdot (P_n(k) + s \cdot P_{fp}(k)) + \frac{s-1}{2} \right).$$

The probabilities that the query of the Bloom filter of the chosen PW in the current node returns a (true) negative, a true positive, and a false positive result as a function of k, the degree of the node holding the resource, are denoted by P_n, P_{tp}, and P_{fp}, respectively.

Proof. We write a recurrence equation for the expected length, given that the search is currently in any of the nodes it visits. Since we have defined the expected search length for any pair of source and target nodes, the expected length of the search from the current node and the expected length of the search from the source node are the same. Denoting it by $\overline{L}_s$, as in the previous section, we can write:

$$\overline{L}_s = (\overline{L}_s + 1) \cdot P_n + (\overline{L}_s + s) \cdot P_{fp} + \frac{s-1}{2} \cdot P_{tp}, \tag{5.5}$$

where P_n, P_{tp}, and P_{fp} are the probabilities that the query of the Bloom filter of the chosen partial walk in the current node returns a (true) negative, a true positive, and a false positive result, respectively, with $P_n + P_{tp} + P_{fp} = 1$. Solving for $\overline{L}_s$, we obtain:

$$\overline{L}_s = \frac{1}{P_{tp}} \cdot (P_n + s \cdot P_{fp}) + \frac{s-1}{2}. \tag{5.6}$$

This equation can be rewritten as:

$$\overline{L}_s = \frac{1 - P_{tp}}{P_{tp}} \cdot \left(\frac{P_n}{1 - P_{tp}} + s \cdot \frac{P_{fp}}{1 - P_{tp}} \right) + \frac{s-1}{2}$$

which is an alternative formulation of the expected search length, in terms of the expected number of partial walks of the search ($\overline{P}$, as defined in Section 5.2.1). Note that $(1 - P_{tp})/P_{tp}$ is the expectation of $\overline{P}$, a geometric random variable representing the number of failures before a Bloom filter returns

a true positive (with probability P_{fp}). The fractions within the parenthesis are, respectively, the probabilities of jumping a partial walk or traversing it, conditional on the fact that the Bloom filter does not return a true positive. Therefore, the terms in the parenthesis are the expectations of $\overline{J}$ and $\overline{U}$, binomial random variables representing the number of jumps and the number of partial walks that are unnecessarily traversed, respectively, as defined in Section 5.2.1.

We now calculate the probabilities in the equations above using $P(i, j)$, the probability that, in the w partial walks of a node, there are i partial walks that contain the node that holds the resource (i.e., their Bloom filters return a true positive), and j partial walks that do not contain the resource, but whose filters return false positives:

$$P(i, j) = B(w, p_r, i) \cdot B(w - i, p, j), \tag{5.7}$$

where $B(m, q, n)$ is the coefficient of the binomial distribution:

$$B(m, q, n) = \left(\begin{array}{c} m \\ n \end{array} \right) \cdot q^n \cdot (1 - q)^{(m-n)}.$$

In Equation 5.7 we are using p_r, defined as the probability that a partial walk includes the node that holds the desired resource. This probability is proportional to the degree of the node that holds the resource, since the probability that a random walk visits a node depends on its degree (see [55], for example). We assume known the number of nodes of each degree k in the network, i.e., its degree distribution, which we denote by n_k.

Denoting by k the degree of the node that holds the resource, the probability that a partial walk of size s contains the resource is then $p_r(k)$, and it can be estimated as:

$$p_r(k) = 1 - \prod_{l=0}^{s-1} \left(1 - \frac{k}{S - l\overline{k}} \right), \tag{5.8}$$

where S denotes the number of endpoints in the network ($S = \sum_k k\, n_k$) and $\overline{k}$ denotes the average degree of the network ($\overline{k} = \sum_k k\, n_k / N$). Each factor in the product in Equation 5.8 represents the probability that the resource is not found in the lth hop of a partial walk, conditional on the fact that it was not found in the previous hops of that partial walk. Note that the fraction $k/(S - l\overline{k})$ is the probability of the lth hop finding the resource, expressed as the number of endpoints that belong to the node that holds the resource divided by the total number of endpoints in the network, except those belonging to nodes already visited by the partial walk, which are $\overline{k}$ per hop, on the average.

Now we rewrite Equation 5.7 making its dependence on k explicit:

$$P(i,j|k) = B(w, p_r(k), i) \cdot B(w-i, p, j),$$

Then, the probabilities in Equations 5.5 and 5.6 are:

$$\begin{aligned} P_{tp}(k) &= \sum_{i=1}^{w} \sum_{j=0}^{w-i} P(i,j|k) \cdot \frac{i}{w} \\ P_{fp}(k) &= \sum_{i=0}^{w} \sum_{j=1}^{w-i} P(i,j|k) \cdot \frac{j}{w} \\ P_n(k) &= 1 - P_{tp}(k) - P_n(k). \end{aligned} \tag{5.9}$$

The expected search length can be finally obtained weighing Equation 5.6 with the probability that the resource is in a node with degree k, which is n_k/N, for all values of k:

$$\overline{L}_s = \frac{1}{N} \sum_k n_k \left(\frac{1}{P_{tp}(k)} \cdot (P_n(k) + s \cdot P_{fp}(k)) + \frac{s-1}{2} \right). \tag{5.10}$$

□

5.3.2. Performance Evaluation

In this section, we compare the analytic results from the model with experimental data from simulations.

Expected Search Length in PW-SAW

Figure 5.12 shows the expected search length ($\overline{L}_s$) as a function of the size of PWs (s) in a regular network, an ER network and a scale-free network, for $p = 0$. The curves in this graph are plotted using Equation 5.10 and previous equations.

According to the results computed using the PW-SAW model, the minimum search lengths occur for values around $s = 141$, $s = 149$ and $s = 167$ for the regular, ER and scale-free networks, respectively. These values are slightly lower than the ones predicted by the PW-RW model (Figure 5.3), which were $s_{opt} = 150, 157$ and 174, respectively.

Both the model curves and the simulation experiments have been computed for $w = 5$, chosen as a reference value. However, it has been observed that very similar results are obtained if we change the value of w. Furthermore, plots of the model equations for different values of w are coincident.

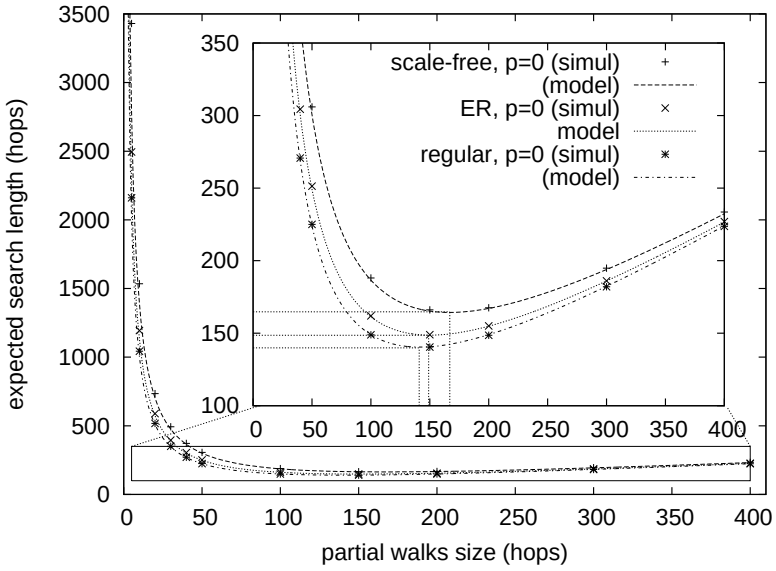


Figure 5.12: Expected search length of PW-SAW in the three networks as a function of s for $p = 0$. The optimal points $(s_{opt}, \overline{L}_{opt})$ for each network are $(141, 139.92)$, $(149, 148.55)$, and $(167, 164.75)$.

This behavior was also observed for PW-RW (Section 5.2.2), where we found that the average search length remained almost constant as we increased w. The reason for this is that the probability of the resource being in the chosen PW (p_r in Equation 5.7) does not depend on the number of PWs in the node.

Experiments have also been performed for other values of the probability of false positive. Figure 5.13 shows the results for a regular network, an ER network and a scale-free network, for $p = 0.01$ and 0.1. Again, the values predicted by the PW-SAW are very close to those obtained experimentally.

We now compare the results of the PW-RW and PW-SAW mechanisms. Figure 5.14 shows results for PW-RW (left part) and for PW-SAW (right part), in the three networks considered in our study, and for values of $p = 0, 0.01$ and 0.1. Expected search lengths from the analytical models are shown as vertical bars, while average search lengths from the simulations experiments are shown as points. The size of the PWs has been set to $s = 150, 157$ and 174 for the regular, ER and scale-free networks, respectively, which are the optimal values predicted by the PW-RW model. For all the networks, we have found a very good correspondence between model predictions and simulation results.

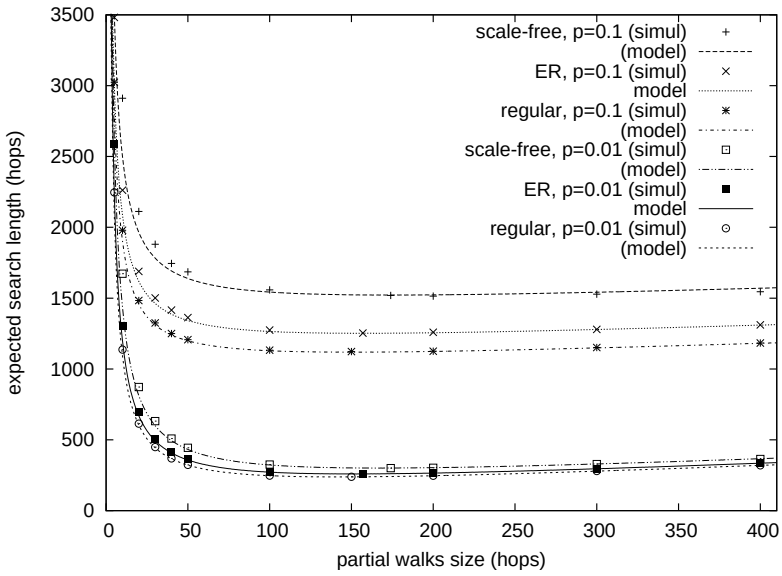


Figure 5.13: Search lengths of PW-SAW as a function of the partial walks length in a regular network, an ER network and a scale-free network for $p = 0.01, 0.1$.

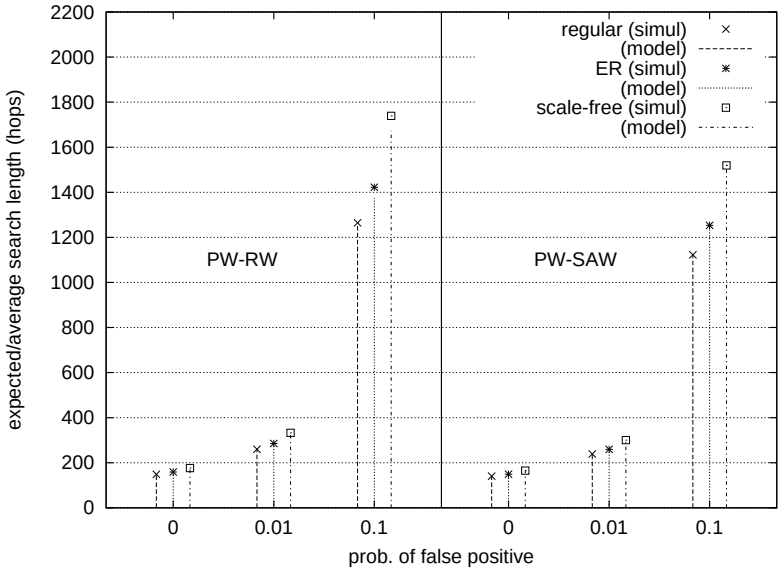


Figure 5.14: Expected search lengths of PW-SAW and PW-RW in the three networks for $p = 0, 0.01, 0.1$.

	Reduction of $\bar{l}$ (%)		
Network type	$p = 0$	$p = 0.01$	$p = 0.1$
Regular	5.67	8.22	11.24
ER	6.25	9.10	11.88
Scale-free	6.53	9.75	12.65

Table 5.2: Reduction of the average search length achieved by PW-SAW with respect to PW-RW.

Comparison of Performance with Respect to Choose-First PW-RW

If we compare the performance of the proposed search mechanisms, we observe that the reduction in the average search length that PW-SAW achieves with respect to PW-RW for a given p is largest for the scale-free network, followed by the ER network and then by the regular network. For each network type, the reduction is larger for higher p. Actual values can be found in Table 5.2.

5.3.3. Alternative Analysis

This section presents an alternative analysis for the model of the choose-first PW-RW mechanism described in Section 5.2.1. This analysis is based on the proof of Theorem 4 for the PW-SAW mechanism. In fact, only the expression for $p_r(k)$ (Equation 5.8), defined as the probability that a given PW contains the node that holds the resource, needs to be rewritten to reflect the fact that the PW is a simple random walk instead of a self-avoiding random walk. The new expression is:

$$p_r(k) = 1 - \left(1 - \frac{k}{S - \overline{k}_{rw}} \cdot \frac{\overline{k}_{rw} - 1}{\overline{k}_{rw}}\right)^s . \tag{5.11}$$

The first fraction within the parenthesis in Equation 5.11 is the ratio of positive endpoints (the degree of the node that holds the resource) and all endpoints in the network ($S = \sum_k k\, n_k$) except those of the current node. We use $\overline{k}_{rw}$, which denotes the expectation of the degree of a node visited by a random walk, as an estimation of the degree of the current node. It can be obtained as:

$$\overline{k}_{rw} = \sum_k k \cdot \frac{k \cdot n_k}{S} = \frac{1}{S} \cdot \sum_k k^2 \cdot n_k.$$

The second fraction within the parenthesis in Equation 5.11 corrects the previous ratio taking into account that, when at a node of a given degree,

the probability of not going backwards (and therefore having the chance to find the resource) is the probability of selecting any of its endpoints but the one that connects it with the node just visited.

The rest of the equations in the proof of Theorem 4 are valid for this alternative analysis of the choose-first PW-RW mechanism.

5.4. Check-First PW-RW and PW-SAW

We now present the check-first versions of the PW-RW and PW-SAW search mechanisms, introduced in Section 5.1.

5.4.1. Analysis

Suppose the search is currently in a node and it needs to pick one of the PWs in that node to decide whether to traverse it or to jump over it. With the new check-first mechanism, it first *checks* the associated resource information of *all* the PWs of the node, and then randomly *chooses* among the PWs with a positive result, if any (otherwise, it chooses among all PWs of the node, as the choose-first version). These check-first mechanisms improve the performance of their choose-first counterparts, since the probability of choosing a PW with the resource increases. This comes at the expense of slightly incrementing the processing power used since several PWs need to be checked, but without incurring extra storage space costs.

A minor additional difference between the algorithms is that in the check-first version, the resource information is registered from the *first* node (the node next to the current node) to the *last* node in the PW. This change slightly improves the performance of the new version, since the probability of choosing a PW with the resource increases also in the cases where the resource is held by the last node of the PW.

We have adapted the analysis presented in the proof for Theorem 4 to reflect the new behavior of the check-first PW-RW/PW-SAW mechanisms. Most of the expressions in the analysis of the choose-first versions are still valid for the check-first versions of the mechanisms, so we present here only the equations that need to be modified to reflect the new behavior. That is the case of Equations 5.9 for the probabilities of choosing a PW with a true positive, false positive, and negative result, respectively. Their counterparts follow. Remember that i and j represent the number of PWs of the node

that return a true positive result and an false positive result, respectively:

$$\begin{aligned} P_{tp} &= \sum_{i=1}^{w} \sum_{j=0}^{w-i} P(i,j) \cdot \frac{i}{i+j}, \\ P_{fp} &= \sum_{i=0}^{w-1} \sum_{j=1}^{w-i} P(i,j) \cdot \frac{j}{i+j}, \\ P_n &= P(0,0) = 1 - P_{tp} - P_{fp}. \end{aligned}$$

The expression for $p_r(k)$ in Equation 5.11 is still valid for check-first PW-RW. However, Equation 5.8 needs to be modified for check-first PW-SAW, since the range of nodes whose resources are associated with the PW has changed from $[0, s-1]$ to $[1, s]$:

$$p_r(k) = 1 - \prod_{l=1}^{s} \left(1 - \frac{k}{S - l\overline{k}}\right).$$

Finally, Equation 5.10 also needs modification (for check-first PW-SAW) in the expectation of trailing steps, for the same reason. The new version, which completes the analysis of the check-first mechanisms, is:

$$\overline{L}_s = \frac{1}{N} \sum_k n_k \left(\frac{1}{P_{tp}(k)} \cdot (P_n(k) + s \cdot P_{fp}(k)) + \frac{s}{2}\right).$$

5.4.2. Performance Evaluation

Figure 5.15 shows the expected search length ($\overline{L}_s$) as a function of the size of PWs (s) in a regular network for the four mechanisms presented so far: choose-first PW-RW/PW-SAW, and check-first PW-RW/PW-SAW, for $p = 0.01$ and $w = 5$. We observe that the check-first mechanisms achieve a lower minimum expected search length than the original choose-first mechanisms, as expected. In fact, the expected search length can be lowered further by increasing w, the number of PWs per node, clearly at the expense of increasing the cost of the PWs construction stage. Also interesting is the observation that the minimum expected search length occurs for significantly lower s (s_{opt} falls from 150 to about 50), meaning shorter PWs in the nodes, which in turn decreases the cost of the PWs construction stage. With regard to the PW-SAW mechanisms, we note that they achieve a slight decrease in the expected search length with respect to the PW-RW mechanisms, for the check-first version as well as for the choose-first version (which was already observed in Table 5.2). Results for the ER and scale-free networks are similar and are omitted here.

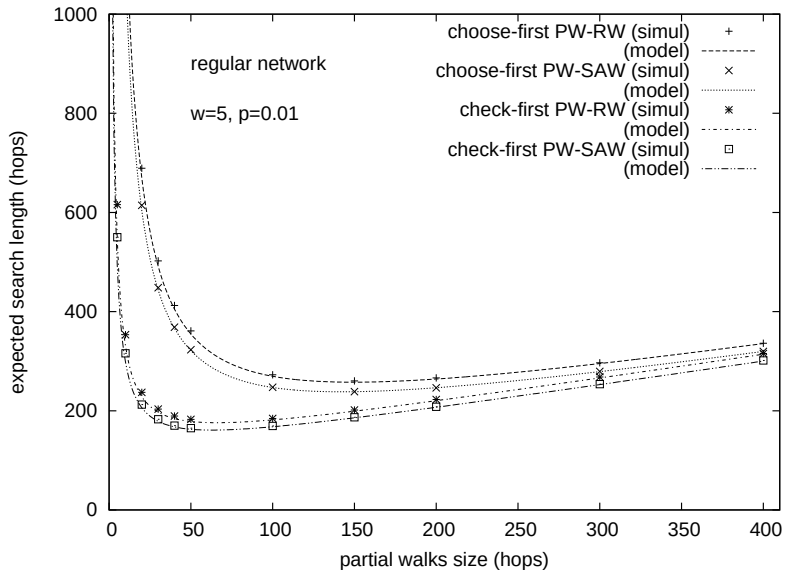


Figure 5.15: Expected search length of choose-first and check-first versions of PW-RW and PW-SAW as a function of s in a regular network for $p = 0.01$ and $w = 5$.

Chapter 6

Partial Random Walks in Networks with Dynamic Resources

This chapter studies the application of the resource location mechanisms based on partial walks (presented in Chapter 5) to networks where resources appear and disappear dynamically. In particular, we focus on the PW-RW mechanism in its two variations (choose-first and check-first), since its reductions in expected search lengths are more relevant than for the PW-SAW mechanism. The dynamic behavior of the resources forces the introduction into the new model of the resource multiplicity feature (the possibility of having several instances of resources, which was not necessary in Chapter 5). In order to isolate the effect of resource dynamics and to assess its impact on the performance of these mechanisms, the use of probabilistic data structures is removed from the model presented in this chapter (i.e., traditional deterministic data structures are used).

Section 6.1 adapts the resource location model presented in Section 5.1 to introduce the new context of dynamic resources and the modifications mentioned above. The analysis of the choose-first is given in Section 6.2.1, and its performance is evaluated in Section 6.2.2. Finally, the analysis and evaluation of the check-first PW-RW mechanism are presented in Sections 6.3.1 and 6.3.2, respectively.

6.1. Model

Let us consider a randomly built network of N nodes and arbitrary topology, with a known degree distribution. Every node holds a set of resources.

We focus on a given resource of interest, of which initially there is a number of instances randomly placed in as many distinct network nodes. Our resource location problem is defined as visiting one of the nodes that hold the resource (the one we encounter first, called the *target node*), starting by a certain node (the *source* node). For each search, the source node is uniformly chosen at random among all nodes in the network. Likewise, we consider that the instances of the resource have been randomly distributed throughout the network. The probability that a given node holds an instance of the resource is denoted by p_{res}. The expectation of the number of instances of the resource for a given network is denoted by $\overline{R} = N \cdot p_{res}$.

Resources have a dynamic behavior. Starting from time T_0, some instances will have disappeared while other new instances will have appeared after an arbitrary observation interval, at time T_r. More concretely, an instance present in a node disappears with probability d. Conversely, an instance not initially present in a node appears in that node with probability a. We will use d as an input parameter to characterize resource dynamics. For a value of d, we will set a so that expectation of the number of resources ($\overline{R}$) remains unchanged. This way, our results will isolate the impact of resource dynamics on the search mechanism due only to the deterioration of information, discarding the effect of a possible increase or decrease in the number of resource instances. Figure 6.1 provides an illustrative example of the dynamic behavior of the resources.

The search mechanism is essentially the one described in Section 5.1 for static resources, with two important modifications: (1) the resource dynamics, and (2) the use of deterministic data structures. We reproduce the entire model here with the necessary modifications.

(1) Partial random walks construction. In time T_0, every node i in the network precomputes a set W_i of w random walks in an initial stage before the searches take place, with the initial distribution of resource instances in the network. Each of these partial walks (PW) has length s, starting at i and finishing at a node reached after s hops. Using the PW-RW mechanism, the PWs computed in this stage are simple random walks.

During the computation of each PW in W_i, node i registers the resources held by the s first nodes in the PW (from i to the one before the last node). The last node of the PW is excluded, being included in the PWs departing from it. The registered information will be used by the searches in stage 2.

(2) The searches. During the interval $(T_0, T_r]$, after the PWs are constructed, searches are performed in the network. We will consider the sys-

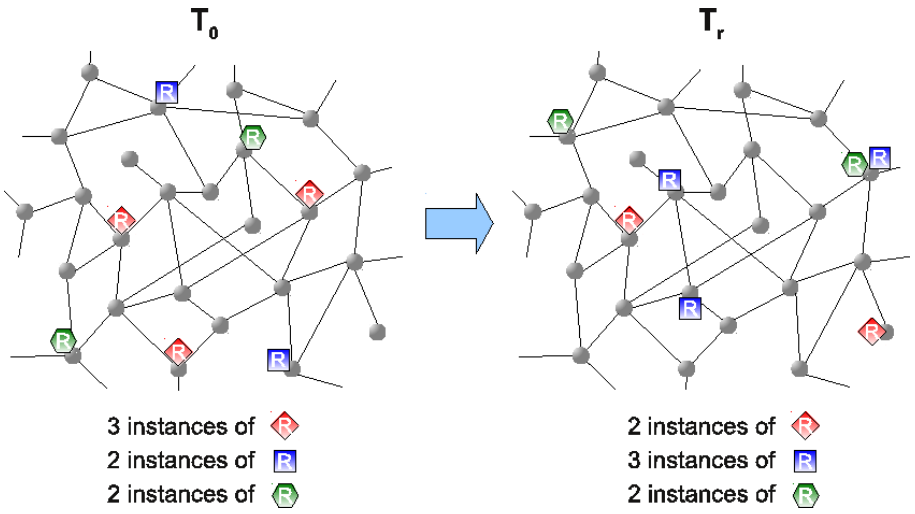


Figure 6.1: Example of the dynamic behavior of resources. There are three resources, each with several instances. It can be seen that such instances change from time instant T_0 to T_r.

tem at time T_r, in which, as stated above, the dynamic behavior of resource instances is characterized by d. Therefore, results obtained will reflect the performance of the search mechanism in a worst case scenario, since searches executed in (T_0, T_r) will see a probability that an instance disappears less than d.

Searches are performed in the following fashion. Let a search start at a node A. A PW in W_A is chosen uniformly at random. Its associated resource information collected in stage 1 is then queried for (any instance of) the desired resource.

- If the result is negative, the search *jumps* to node B, the last node of that PW.

  The process is then repeated at B and the search keeps jumping in this way while the results of the queries are negative.

- If, when at a node C, the query returns a positive result, the search *traverses* that PW looking for the resource. It starts checking if the current node has the desired resource. If it does not, the search takes a *step* to the next node of the PW, checking again if it has the resource. The search proceeds through the PW in this way until the resource is found or the PW is finished.

If the resource is found, the search stops. Otherwise (i.e., if the search reaches the last node of the PW without having found the resource in the previous nodes), it means that the information collected in stage 1 for that PW and the resource of interest is no longer valid. The search then considers that the result is negative, and the search process is repeated at the last node of the PW.

As for the resource location mechanisms presented in Chapter 5, these mechanisms shorten search lengths because of the number of steps saved in jumps over PWs in which we know that the resource is not located, although these saving may be reduced by the unnecessary steps due, in this case, to *outdated information due to the resources dynamic behavior.*

Resource Dynamics

Regarding resource dynamics, we realize that searches are executed based on information collected in time T_0 that may be outdated in T_r, when the queries are performed. Four cases arise when the information associated with a PW is queried for the resource, as is shown in Table 6.1 and illustrated by Figure 6.2.

- **True Negative:** A *True Negative* or TN (case a) in Figure 6.2) occurs when no instances were present in the PW at T_0, and the same holds at T_r.
- **True Positive:** A *True Positive* or TP (case b) occurs when one or more instances are present in the PW at T_0 and one or more instances (not necessarily the same ones) are present at T_r.

  The impact of resource dynamics on the performance of the search mechanism comes from the *False Negatives* and the *False Positives*:

- **False Negative:** A *False Negative* or FN (case c) occurs when there were no instances in the PW at T_0, but at least one instance is present at T_r. It makes the search jump over that PW, ignoring the new instance(s).
- **False Positive:** A *False Positive* or FP (case d) occurs when there were one or more instances in that PW at T_0 but all of them are gone at T_r *and* there are no new instances at T_r. It makes the search traverse a whole PW fruitlessly, since no instances are currently in that PW. Note that the case when all instances disappear from the PW, but some other instance(s) appears in that PW is included in the TP case.

	Resource present in PW			
case	T_0	T_r	query result	
a)	no	no	True Negative	(TN)
b)	yes	yes	True Positive	(TP)
c)	no	yes	False Negative	(FN)
d)	yes	no	False Positive	(FP)

Table 6.1: Resource dynamics: query results.

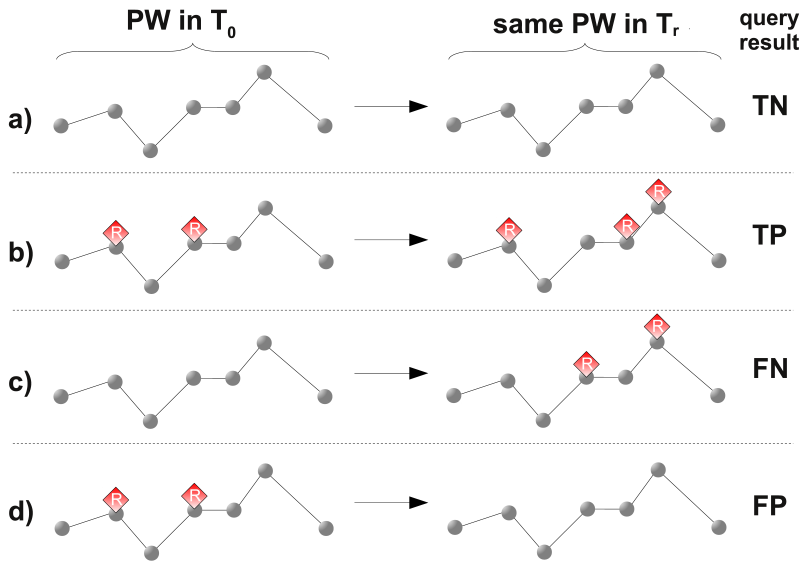


Figure 6.2: Resource dynamics: information on resources in a PW is gathered at T_0; queries are performed at T_r.

6.2. Choose-First PW-RW

This section presents the analysis and performance evaluation of the choose-first PW-RW mechanism, according to the model described in the previous section.

6.2.1. Analysis

As in previous chapters, our objective is to come to an analytical expression for the expected search length of the PW-RW mechanisms. The analysis of choose-first PW-RW in networks with dynamic resources and deterministic data structures is an adaptation of the alternative analysis of choose-first PW-RW for static networks and with probabilistic data structures (Bloom filters) given in Section 5.3.3, which in turn was based on the proof of Theorem 4 (Section 5.3.1).

We start from a recurrence equation for the expected length, given that the search is currently in any of the nodes it visits. Since we have defined the expected search length for any pair of source and target nodes, the expected length of the search from the current node and the expected length of the search from the source node are the same. Denoting by $\overline{L}_s$ the expected search length when using PWs of length s, as in Chapter 5, we can write:

$$\overline{L}_s = (\overline{L}_s + 1) \cdot P_n + (\overline{L}_s + s) \cdot P_{fp} + \overline{T} \cdot P_{tp}, \tag{6.1}$$

where P_n, P_{tp}, and P_{fp} are the probabilities of choosing a partial walk (out of the w PWs of the node) for which the query for the resource returns a negative (either a TN or a FN, see Table 6.1), a TP, and a FP result, respectively, with $P_n + P_{tp} + P_{fp} = 1$. $\overline{T}$ is the expectation of the number of trailing steps taken when traversing the last PW, until an instance of the resource is found (TP). Note that $\overline{T}$ is no longer $\frac{s-1}{2}$ as in Section 5.3.1, since there can be more than one resource instance in the final PW. An expression for $\overline{T}$ will be obtained later. Solving for $\overline{L}_s$, we obtain:

$$\overline{L}_s = \frac{1}{P_{tp}} \cdot (P_n + s \cdot P_{fp}) + \overline{T}. \tag{6.2}$$

The probabilities in Equation 6.2 are estimated with the following expressions:

$$
\begin{aligned}
P_{tp} &= \sum_{i=1}^{w} \sum_{j=0}^{w-i} P(i,j) \cdot \frac{i}{w}, \\
P_{fp} &= \sum_{i=0}^{w-1} \sum_{j=1}^{w-i} P(i,j) \cdot \frac{j}{w}, \\
P_{n} &= \sum_{i=0}^{w-1} \sum_{j=0}^{w-i-1} P(i,j) \cdot \frac{w-(i+j)}{w} = 1 - P_{tp} - P_{fp}, \qquad (6.3)
\end{aligned}
$$

where $P(i,j)$ is the probability that, in the w PWs of a node, there are i PWs whose queries return a TP result and j PWs that return a FP result:

$$P(i,j) = B(w, p_{tp}, i) \cdot B(w-i, p_{fp}, j),$$

where

- $B(m,q,n)$ is the coefficient of the binomial distribution

$$B(m,q,n) = \left(\begin{array}{c} m \\ n \end{array} \right) \cdot q^n \cdot (1-q)^{(m-n)},$$

- p_{tp} is the probability that a given PW at any node returns a TP result[1],
- and p_{fp} is the probability that a given PW at any node returns a FP result, conditioned on the fact that it does not return a TP.

Therefore, in order to evaluate the estimation of the expected search length given by Equation 6.2, we need to obtain the values of p_{tp}, p_{fp} and $\overline{T}$. Let us provide them:

1. The variable p_{tp} has been defined as the probability that a given PW at any node returns a TP result. This probability can be easily estimated if we condition it on the fact that the PW (of length s) had exactly r instances of the resource at T_0. Defining $P_{pw}(r)$ as the probability that a PW has r instances of the resource, and recalling that $\overline{R}$ is the expectation of the number of instances of the resource in the network, we can write:

[1]This probability p_{tp} is not to be confused with P_{tp} (note the different case), defined above as the probability of *choosing* a PW which returns a TP out of the w PWs of the current node.

$$p_{tp} = \sum_{r=1}^{\min\{s,\overline{R}\}} P_{pw}(r) \cdot \left[(1-d^r) + d^r \cdot \left(1-(1-a)^{s-r}\right)\right],$$

where the brackets contain the probability that not all the r instances present at T_0 have disappeared (with probability d) at T_r or, if they did disappear, at least one instance appeared (with probability a) in some of the $s-r$ remaining nodes in that interval.

An estimation for $P_{pw}(r)$ can be obtained using the random properties of a random walk in networks built randomly. In particular, we consider that the next hop of a random RW can take it to any of the endpoints in the network (except the endpoints of the current node since we do not allow self-loops). Then we estimate $P_{pw}(r)$ as $B(s, p_{rw}, r)$, where p_{rw} is the probability that the RW visits a node with an instance of the resource in the next hop. In turn, we estimate this probability as:

$$p_{rw} = \frac{\overline{R} \cdot \overline{k}}{S - \overline{k}_{rw}} \cdot \frac{\overline{k}_{rw} - 1}{\overline{k}_{rw}}. \tag{6.4}$$

The first fraction in Equation 6.4 is the ratio of positive endpoints (the ones connected to the $\overline{R}$ nodes that have an instance of the resource) and all endpoints in the network ($S = \sum_k k\, n_k$) except those of the current node. We use the average degree of the network ($\overline{k} = \sum_k k\, n_k/N$) as an estimation of the degree of a node that holds the resource (which is assigned or not with uniform probability p_{res} across the network). Similarly, we use the expectation of the degree of a node visited by a random walk as an estimation of the degree of the current node:

$$\overline{k}_{rw} = \sum_k k \cdot \frac{k \cdot n_k}{S} = \frac{1}{S} \cdot \sum_k k^2 \cdot n_k.$$

The second fraction in Equation 6.4 corrects the previous ratio taking into account that, when at a node of a given degree, the probability of not going backwards (and therefore having the chance to find the resource) is the probability of selecting any of its endpoints but the one that connects it with the node just visited by the walk. With this, the estimation of $P_{pw}(r)$ is:

$$P_{pw}(r) = \left(\begin{array}{c} s \\ r \end{array} \right) \cdot (p_{rw})^r \cdot (1-p_{rw})^{s-r}.$$

2. We have defined p_{fp} as the probability that a given PW at any node returns a FP result, conditioned on the fact that it does not return a TP. This conditioning comes from the second binomial coefficient in this equation, which we restrict to the $w - i$ PWs which we know that do not return a TP, since the ones that do are accounted for in the first binomial coefficient. In other words, the second binomial coefficient includes the PWs that return a TN, a FN or a FP result, and p_{fp} is the probability that it returns a FP conditioned on that. We can then easily write an estimation of p_{fp} as:

$$p_{fp} = \frac{1}{1 - p_{tp}} \left(1 - P_{pw}(0) - p_{tp}\right),$$

where we are subtracting $P_{pw}(0)$ (the probability of cases TN and FN in Table 6.1), and p_{tp} (the probability of case TP).

3. An expression for $\overline{T}$, the expectation of the number of trailing steps taken when traversing the last PW until an instance of the resource is found, is still needed to be finally able to estimate the average search length in Equation 6.2. For this we rely on $\overline{T}(r)$, the expectation of that variable conditioned on there being r instances of the resource in the PW. Then:

$$\overline{T} = \frac{1}{1 - P_{pw}(0)} \cdot \sum_{r=1}^{\min\{s,\overline{R}\}} \overline{T}(r) \cdot P_{pw}(r). \tag{6.5}$$

Note that $\overline{T}$ is in fact conditioned on there being at least one instance of the resource in the PW, since it corresponds to a TP (see Equation 6.1). This is the reason of the fraction multiplying the summation in Equation 6.5.

Now we provide an expression for $\overline{T}(r)$ as the expectation of the position of the first resource in the PW (conditioned on there being r instances of the resource in the PW):

$$\overline{T}(r) = \sum_{i=0}^{s-r} \left[i \cdot \left(\prod_{j=0}^{i-1} \left(1 - \frac{r}{s-j}\right) \right) \cdot \left(\frac{r}{s-i} \right) \right]. \tag{6.6}$$

Each factor in the product of Equation 6.6 is the probability that there is no instance in the j_{th} position of the PW, conditioned on that there is no instance in the previous position. The final factor outside the

product is the probability that there is an instance in the i_{th} position conditioned on there are no instances in the previous positions.

Cost of Precomputing PWs

Let's consider the interval $(T_0, T_r]$, chosen so that the probability d at T_0 has some acceptable value.[2] Searches performed in this interval use the PWs precomputed at T_0, and thus the cost of this computation must be added to the cost of the searches themselves. The analysis of this cost is the same as in Section 5.2.1. We reproduce here the expression of C_t, the total cost in terms of number of messages. This cost includes the precomputation of PWs and the searches themselves:

$$C_t = (\overline{L}_s + 1) + \frac{w}{b}(s + 1),$$

as a function of b, the average number of searches started by nodes in the interval $(T_0, T_r]$.

6.2.2. Performance Evaluation

The goal of this section is to apply the analytical model presented in the previous section to real networks, and to validate its predictions with data obtained from simulations. The experimental framework described in Section 3.5 is used. As in previous chapters, three types of networks have been chosen for the experiments: regular networks, Erdős-Rényi (ER) networks and scale-free networks (see Section 3.3.4). The experiments settings are similar to the ones in Chapter 5: a network of each type and size $N = 10^4$ has been randomly built with the method proposed by Newman et al. [67] for networks with arbitrary degree distribution, setting their average node degree to $\overline{k} = 10$. For each experiment, 10^6 searches have been performed. In every search, the source node has been chosen uniformly at random, and every node in the network has been assigned an instance of the resource looked for with probability $p_{res} = 10^{-2}$ at T_0. Therefore, the expected number of resource instances present in the network at T_0 for all searches is $\overline{R} = N \cdot p_{res} = 100$.

[2]The interval for which the dynamic behavior of resources yields a given value of d depends on the stochastic processes that governs the births and deaths of resource instances in the nodes of the network. The determination of that interval is out of the scope of this work.

Expected search lengths

Figures 6.3 to 6.5 show the expected search length in a regular, ER and scale-free network, respectively, for several values of d. The number of PWs per node is set to $w = 5$, although the performance of the PW-RW mechanism is independent from this parameter[3], since only one PW is used to locate the resource. On the contrary, this parameter plays a central role in the check-first PW-RW, whose performance results are shown in Section 6.3.2. Expected search lengths predicted by the model (Section 6.1) are plotted as curves and simulation results (average search lengths) are shown as points. It can be seen that the model provides an accurate approximation of the real data, with larger error for higher values of d and s. Among the network types, the regular network shows the smallest errors, followed by the ER network and the scale-free network. These discrepancies are accounted for by the greater dispersion of node degree in the ER, and especially, in the scale-free networks. Recall from Section 6.1 that in our model we are using the average degree of the network ($\overline{k}$) as an estimation of the degree of the nodes that hold the resource, and the expectation of the degree of a node visited by a random walk as an estimation of the degree of the current node ($\overline{k}_{rw}$). These estimations grow less accurate as the dispersion of the node degree increases. Predictions of $\overline{L}_s$ are slightly lower than experimental data, rendering an optimistic model in general, with a very good fit for interesting values of s and values of d not large.

All curves show a minimum point, which marks the optimal PW length (s_{opt}) and the corresponding optimal expected search length ($\overline{L}_{opt}$). Interestingly, the values of s_{opt} are small and do not depend heavily on d or on the network type. According to the analytic data, s_{opt} ranges between 13 and 15 for all curves shown and for the three network types. Values of $\overline{L}_{opt}$ are also very similar. For example, for $d = 0.3$, $\overline{L}_{opt} = 21.95$ (regular), 21.83 (ER), and 21.32 (scale-free).

Search Length Distributions

The use of partial walks also affects the shape of the probabilistic distribution of search lengths. Figure 6.6 shows the distributions for simple random walks (RW) and for PW-RW, for $s = 10$ and for several values of d, obtained from the experiments with the regular network. Instead of the slowly decaying distribution of random walks, the proposed mechanism exhibits search length distributions that show a maximum frequency for a small

[3]As observed in Section 5.2.2, the extreme case of having just *one* PW is to be avoided because it produces loops that cause many unfinished searches.

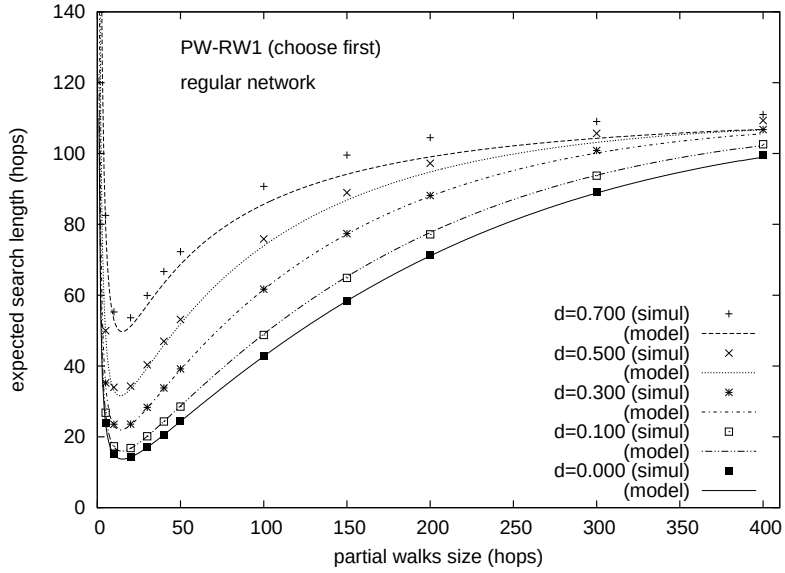


Figure 6.3: Choose-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a regular network with $w = 5$.

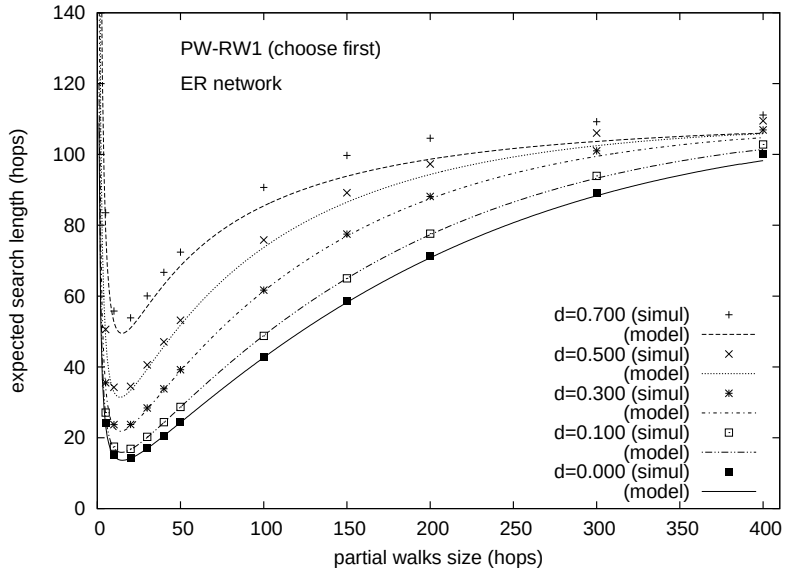


Figure 6.4: Choose-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a ER network with $w = 5$.

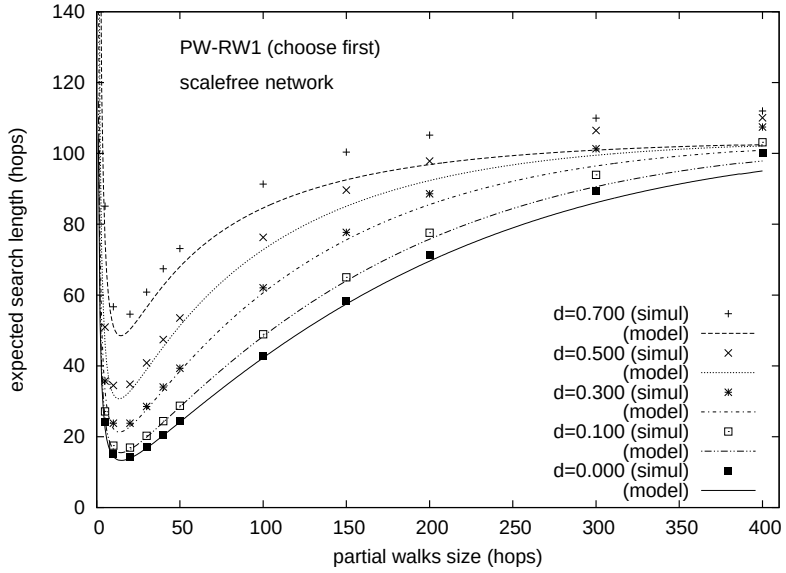


Figure 6.5: Choose-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a scale-free network with $w = 5$.

search length and then decay much faster than the random walk distribution. We also note that the search length for the maximum frequency (9 in this case) is independent from the dynamic behavior of resources (d). The search length distributions of PW-RW have therefore lower standard deviation (and also lower coefficient of variation) than the random walk searches. Regarding the ER and the scale-free networks, their search length distributions have similar shape, mean and standard deviation values, and are not shown here.

Comparison with RW Searches

We have compared the performance of the proposed search mechanism for $\overline{L}_{opt}$ with searches based on simple random walks. Table 6.2 shows the relative reductions (%) for several values of d for the network types we have considered. We can see that the reduction in the average search length that PW-RW achieves with respect to simple random walk is lower for higher d, ranging from around 87% in the case when $d = 0$ to 55% when $d = 0.7$. Furthermore, we also see that the achieved reductions are independent of the network type, with small differences.

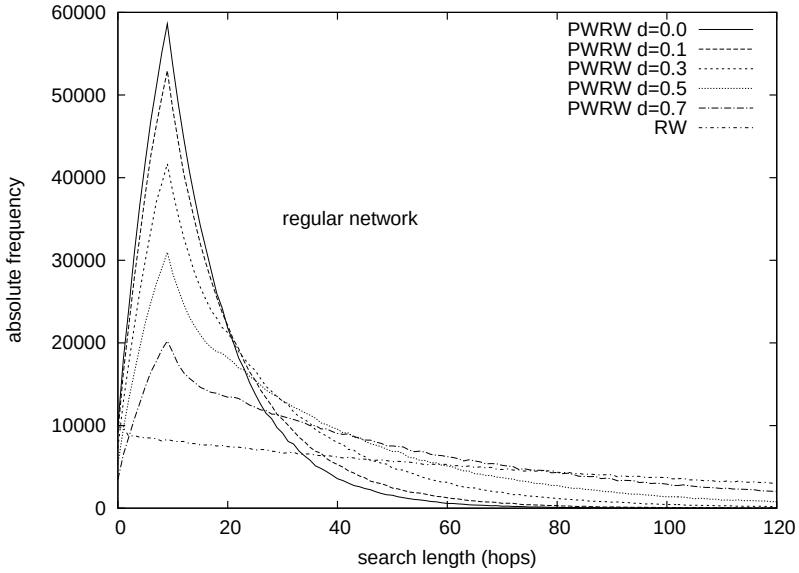


Figure 6.6: Search length distributions for RW searches and PW-RW searches, for a regular network with $s = 10$ and $d = 0, 0.1, 0.3, 0.5$ and 0.7.

d	Regular	ER	Scale-free
0.0	87.73 %	87.87 %	88.26 %
0.1	85.76 %	85.87 %	86.36 %
0.3	80.24 %	80.57 %	81.22 %
0.5	71.82 %	72.02 %	72.93 %
0.7	55.68 %	55.95 %	57.25 %

Table 6.2: Reduction of the expected search lengths of PW-RW relative to random walk searches for several d.

6.3. Check-First PW-RW

This section presents the analysis and performance evaluation of the check-first PW-RW mechanism, according to the model described in Section 6.1.

6.3.1. Analysis

The analysis of check-first PW-RW in networks with dynamic resources and deterministic data structures is based on the one provided in Section 5.4.1 for static resources and probabilistic data structures (Bloom filters). The considerations made in that section are applicable to the new context. In addition, most of the analysis provided for choose-first PW-RW in Section 6.2.1 is still valid for check-first PW-RW. We present here the equations that need to be modified to reflect the new behavior. That is the case of Equations 6.3 for the probabilities of choosing a PW with a TP, FP and negative result, respectively. Their counterparts have been modified in the same way that in Section 5.4.1. Remember that i and j represent the number of PWs of the node that return a TP result and a FP result, respectively:

$$\begin{aligned} P_{tp} &= \sum_{i=1}^{w}\sum_{j=0}^{w-i} P(i,j)\cdot\frac{i}{i+j}, \\ P_{fp} &= \sum_{i=0}^{w-1}\sum_{j=1}^{w-i} P(i,j)\cdot\frac{j}{i+j}, \\ P_{n} &= P(0,0) = 1 - P_{tp} - P_{fp}, \end{aligned}$$

The expression for the expectation of the number of trailing steps taken when traversing the last PW until the resource is found (Equation 6.5) is still valid. It uses $\overline{T}(r)$, the expectation of the position of the first resource in the PW, conditioned on there being r instances of the resource in the PW. Its expression (Equation 6.6) needs to be modified, since the range of nodes whose resources are associated with the PW has changed from $[0, s-1]$ to $[1, s]$. The indexes limits and their use in the expression have been updated as necessary in the new expression, which completes the analysis of the check-first PW-RW mechanism:

$$\overline{T}(r) = \sum_{i=1}^{s-r+1}\left[i\cdot\left(\prod_{j=1}^{i-1}\left(1-\frac{r}{s-j+1}\right)\right)\cdot\left(\frac{r}{s-i+1}\right)\right].$$

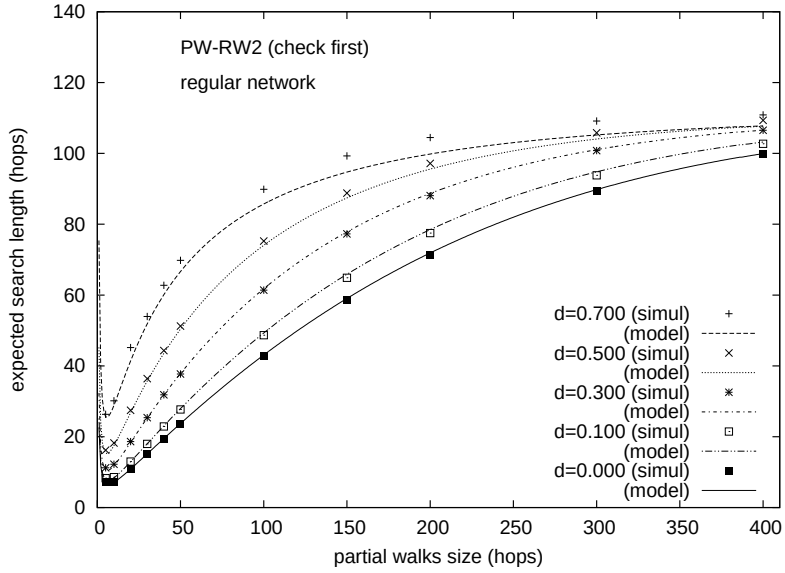


Figure 6.7: Check-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a regular network with $w = 5$.

6.3.2. Performance Evaluation

In this section we present the performance of the check-first variation of the PW-RW mechanism described and analyzed in the previous section. Figures 6.7 to 6.9 show the expected search length in a regular, ER and scale-free network, respectively, for several values of d, and for $w = 5$. The shape of the curves is the same as that for the original mechanism (Figures 6.3 to 6.5), with a substantial decrease in the optimal search length ($\overline{L}_{opt}$). For example, for $d = 0.3$, $\overline{L}_{opt}$ is around 11 for the three network types, while it was about 21 for the PW-RW mechanism. The optimal PW length also diminishes, from about 14 to 6 in that case. The expected search length decrease is due to the fact that the new mechanism checks all the PWs in the node for the resource and then chooses one only among those with positive result, increasing the probability of choosing a PW that currently holds the resource. Following this reasoning, the more PWs in the node, the higher this probability. It is therefore interesting to explore the dependency of $\overline{L}$ and s_{opt} with w.

Figure 6.10 shows the expected search length of the check-first PW-RW mechanism, for $w = 2, 5$ and 10 in a regular network with $d = 0.3$. To make the comparison of performance between both mechanisms easier, a curve corresponding to choose-first PW-RW has been added to the graph. As explained above, the performance of the latter is independent from w. The range of the axes of this graph has been restricted to focus on the

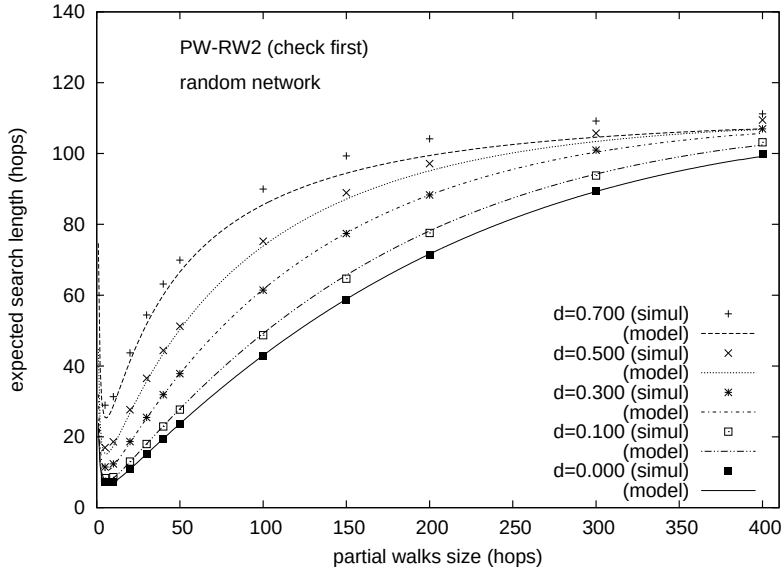


Figure 6.8: Check-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a ER network with $w = 5$.

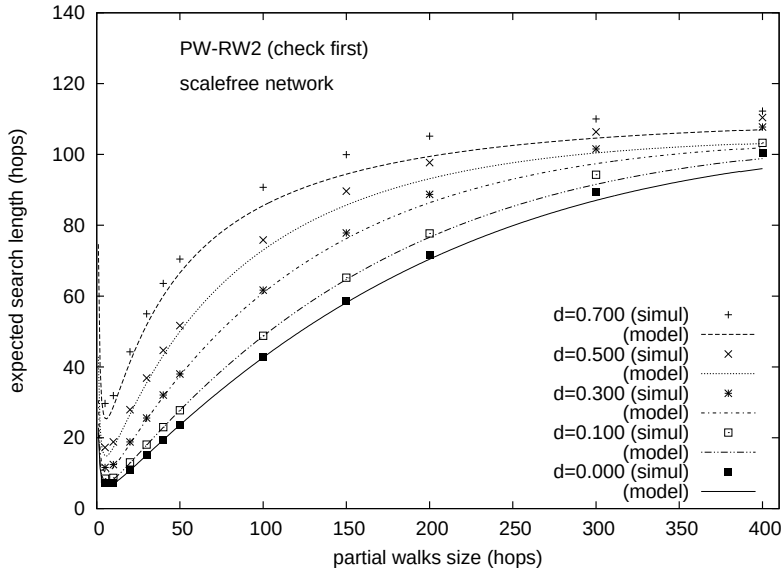


Figure 6.9: Check-first PW-RW. Expected/Average search length ($\overline{L}_s$) vs. PW length (s) for $d = 0.0, 0.1, 0.3, 0.5, 0.7$ in a scale-free network with $w = 5$.

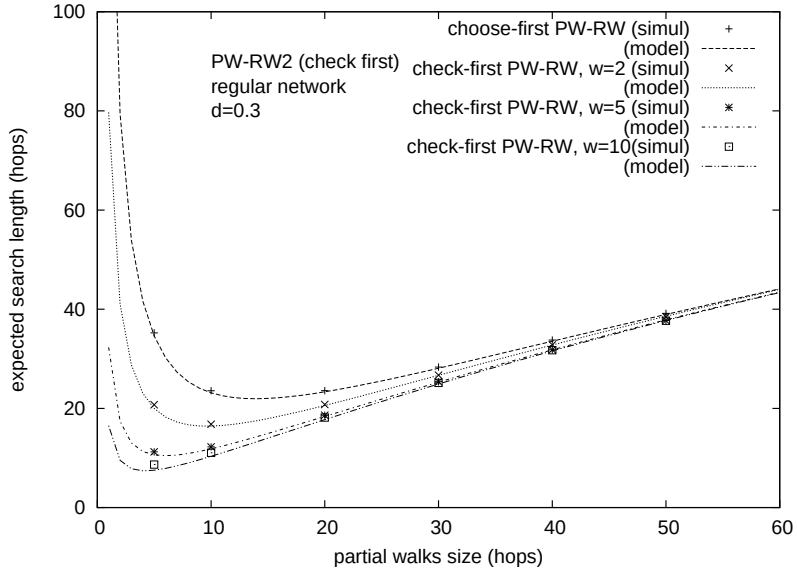


Figure 6.10: Search lengths ($\overline{L}_s$) vs. PW length (s) for check-first PW-RW with $w = 2, 5, 10$, and choose-first PW-RW in a regular network with $d = 0.3$.

d	Regular	ER	Scale-free
0.0	94.02 %	94.11 %	94.29 %
0.1	93.09 %	93.20 %	93.43 %
0.3	90.59 %	90.74 %	91.04 %
0.5	86.27 %	86.49 %	86.91 %
0.7	77.08 %	77.43 %	78.12 %

Table 6.3: Reduction of the expected search lengths of check-first PW-RW ($w = 5$) relative to random walk searches for several d.

area around s_{opt}. For higher s, the curves for the several w converge. As expected, it is observed that higher w yields lower $\overline{L}_{opt}$, with a value about 8 for $w = 10$. Another interesting observation is that s_{opt} also diminishes for higher w, falling to 4 in this case. These values mean a reduction of about 92% in the expected search length of simple random walks, with 10 precomputed PWs of just 4 nodes. Higher reductions can be achieved, at the expense of increasing the cost of the computation of the PWs, as analyzed in Section 6.2.1.

Results for the ER and scale-free networks are similar and the graphs corresponding to Figure 6.10 are not shown here. Table 6.3 is provided as a reference, presenting the reductions achieved by check-first PW-RW for $w = 5$ with respect to random walk searches in a regular, ER and scale-free

networks for several d. We see that reductions range between 94% and 77%, while those of choose-first PW-RW ranged between 88% and 55%.

Chapter 7

Conclusions and Future Work

This dissertation joins the efforts of the research community to improve the performance of the operation of complex networks, particularly in what concerns the location of resources. Complex networks may be large, randomly built, and dynamic. They include, but are not limited to, P2P systems, ad-hoc wireless networks, contents networks, social networks, etc. Because of the mentioned characteristics, knowledge about their topology and other attributes cannot be used reliably to locate resources, or it may be simply unavailable. This makes the provision of central services difficult or not desirable. Therefore, we have tried to devise resource location mechanisms that use as little information about the network as possible, and without any impact on the network structure.

Our focus has consequently been on decentralized unstructured systems, since they do not impose an overlay network structure and a resource placement strategy to facilitate the location of resources. Resource location mechanisms for structured systems do provide efficient searches, but at the cost of setting up and maintaining the resource placement scheme in the face of a possibly changing network. The mechanisms we have proposed do not need any particular network structure, and they use only local information. They use random walks as a compromise solution between the inefficient flooding on one hand, and the supernodes systems and centralized systems on the other, which in fact cross the limits of the peer-to-peer paradigm.

7.1. Summary of Results

We have modelled, analyzed and evaluated the performance of a family of resource location mechanisms. The analysis has been directed to obtain the expected search lengths achieved by each mechanism. Their predictions

have been checked against experimental data from simulations, particularly in three types of randomly built networks: regular, Erdős-Rényi, and scale-free. Considered networks are random also in the sense that a link departing from a given attachment point of a node can finish in any other attachment point of the network with uniform probability. Both the analysis and the network building procedure used in the simulations take in this property. Simulations have shown that expected search lengths from the analytical models accurately estimate the average search lengths obtained from real data.

First, we have considered the use of self-avoiding random walks (SAW) for resource location. Our main contribution has been an analytical mean-field model to estimate the expected search length in networks characterized only by their size and degree distribution. The model considers the possible use of one-hop resource replication and the existence of multiple resource instances.

We have compared the performance of SAW and RW. The comparison of analytical results have required the extension of an existing model for RW searches to include the features mentioned above. Experiments have been performed for the two mechanisms. SAW achieves important reductions of (expected/average) search lengths in networks without one-hop replication, ranging between 50% and 75%, depending on the network type and its average degree. Scale-free networks exhibit the larger reductions, which increase as the average degree of the network grows (as opposed to regular and ER networks). In networks with one-hop replication, reductions go down to range between 7% and 26%, showing that the benefits of avoiding revisits are partially compensated by the knowledge of the resources of the neighbors. Conclusions about the impact of network size, type, average degree, one-hop replication, and resource instances on the search lengths have also been drawn in our performance study.

The next resource location mechanisms we have proposed are based on the idea of connecting precomputed short random walks (partial walks or PW) to obtain longer walks to be used in more efficient searches. The main idea is to collect information on the resources held by the nodes when the partial walks are constructed. This information can be then used when performing searches, making it possible to jump over partial walks where we know the resource is not located, saving many search hops. Probabilistic data structures (e.g. Bloom filters) have been introduced in case the storage of the full information is too costly. However, they introduce a probability of having false positives when querying a PW for a resource, thus decreasing the benefits

from jumping over PWs. Partial walks can be either RW or SAW, resulting in the PW-RW and PW-SAW mechanisms. Two variations (choose-first and check-first) of these mechanisms are defined, depending on whether a PW in the current node is chosen first and then checked for the desired resource, or all its PWs are checked and then one is chosen.

We have developed analytical models for these mechanisms and performed simulation experiments to validate their predictions. Choose-first PW-RW achieves an expected search length proportional to the square root of that obtained by simple random walk searches, when the probability of a false positive is small. The model also gives the optimal length of PWs, i.e., it produces the minimum expected search length. Reductions of search lengths with respect to RW searches range between 88% and 98% in the experiments performed, with no strong dependency on the network type, and decreasing for growing probabilities of false positives. We have found that just two PWs per node are enough to obtain a statistical behavior similar to that of a true random walk built with PW that are never reused.

Choose-first PW-SAW achieves further reductions of the expected search length (between 5% and 12%), this time increasing with the false positive probability. The contribution of SAW with respect to RW is not as large as in the previous mechanism, since PWs are short and the number of new nodes visited is close to the number of hops of the walk (in large networks with relatively high average degree, as in our experiments).

Check-first versions of the mechanisms provide additional reductions, which increase as we use more PWs per node, clearly to the expense of incrementing the cost of precomputing PWs. Increasing the number of PWs per node also has the benefit of reducing the length of the optimal PWs.

Finally, we have adapted the PW-RW mechanisms to networks where resources have a dynamic behavior, appearing and disappearing from the nodes. This causes that the information collected about resources when precomputing PWs to deteriorate with time, introducing false negatives and false positives when querying a PW for the desired resource. Only deterministic data structures are used in this new scenario, to isolate the effect of the resource dynamics.

We have presented analytical models for choose-first and check-first mechanisms in the new scenario. We have found that the large reductions of the search lengths that choose-first PW-RW achieves compared to RW searches are reduced by the resource dynamics, but they remain significant even in the face of high volatility of resources. Check-first PW-RW produces larger reductions when the number of PWs per node is increased, as it was observed

in the previous scenario for static resources.

7.2. Thesis

The results and considerations in the previous section allow us to conclude that the hypothesis stated in Section 1.3 has become a thesis:

Thesis: Variations of random walks or combinations with other mechanisms can be used to devise efficient resource location mechanisms that outperform simple random walk searches in terms of the average search length.

In particular, this is true for the mechanisms based on self-avoiding random walks and on partial walks described in this dissertation, applied to randomly built networks where any link may be connected to any other attachment point in the network with uniform probability.

7.3. Future Work

This work can be continued along several research lines to obtain more efficient resource location mechanisms:

- Modify the random walk algorithm in the rules used to choose the neighbor to be visited next. SAW chooses, if possible, a non-visited neighbor. Other strategies could take into consideration the degree of the neighbors, the number of times they have already been visited or covered, the number of resources held, etc.

- Modify the criteria used to choose a partial walk from the ones available at the current node. For example, if no partial walk holds the desired resource, the partial walk with higher average degree of its nodes could be selected.

- Analyze and evaluate the performance of the PW mechanisms when both probabilistic data structures and the dynamics of resources are combined as sources of false positives.

- Analyze and evaluate the performance of the mechanisms in networks with dynamic behavior of nodes, i.e., with nodes churn. An complementary approach would be to consider link dynamics, i.e. nodes remain in the network but their links with other nodes change over time.

- Finally, it would be interesting to make a detailed comparison study of the benefits and costs of the proposed mechanisms and of approaches used in structured systems (e.g., DHTs).

Bibliography

[1] Lada A Adamic and Bernardo A Huberman. Zipf's law and the Internet. *Glottometrics*, 3(1):143–150, 2002.

[2] Lada A Adamic, Rajan M Lukose, Amit R Puniyani, and Bernardo A Huberman. Search in power-law networks. *Physical Review E*, 64(4):046135, 2001.

[3] Murat Alanyali, Venkatesh Saligrama, and O Sava. A random-walk model for distributed computation in energy-limited network. In *Proceedings of 1st Workshop on Information Theory and its Application (2006)*, 2006.

[4] David J Aldous. Lower bounds for covering times for reversible Markov chains and random walks on graphs. *Journal of Theoretical Probability*, 2(1):91–100, 1989.

[5] Daniel J Amit, G Parisi, and L Peliti. Asymptotic behavior of the "true" self-avoiding walk. *Physical Review B*, 27(3):1635, 1983.

[6] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004.

[7] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific American*, 2003.

[8] Greg Barnes and Uriel Feige. Short random walks on graphs. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (1993)*, pages 728–737. ACM, 1993.

[9] B Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (2003)*, pages 49–60. IEEE, 2003.

[10] Nabhendra Bisnik and Alhussein Abouzeid. Modeling and analysis of random walk search algorithms in P2P networks. In *Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems, 2005 (HOT-P2P 2005)*, pages 95–103. IEEE, 2005.

[11] Website of BitTorrent. http://www.bittorrent.com.

[12] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[13] Graham Brightwell and Peter Winkler. Maximum hitting time for random walks on graphs. *Random Structures & Algorithms*, 1(3):263–276, 1990.

[14] Andrei Broder and Michael Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[15] Julián Candia, Paul E Parris, and VM Kenkre. Transport properties of random walks on scale-free/regular-lattice hybrid networks. *Journal of Statistical Physics*, 129(2):323–333, 2007.

[16] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making Gnutella-like P2P systems scalable. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (2003)*, pages 407–418. ACM, 2003.

[17] Yan Chen, Randy H Katz, and John D Kubiatowicz. Scan: A dynamic, scalable, and efficient content distribution network. In *Pervasive Computing*, pages 282–296. Springer, 2002.

[18] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient search in unstructured peer-to-peer networks. *European Transactions on Telecommunications*, 15(6):535–548, 2004.

[19] Ian Clarke, Scott G Miller, Theodore W Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with FreeNet. *Internet Computing, IEEE*, 6(1):40–49, 2002.

[20] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[21] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 177–190. ACM, 2002.

[22] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (2002)*, pages 23–32. IEEE, 2002.

[23] Luciano da Fontoura Costa and Gonzalo Travieso. Exploring complex networks through random walks. *Physical Review E*, 75(1):016102, 2007.

[24] Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. Efficient distributed random walks with applications. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC'10)*, pages 201–210. ACM, 2010.

[25] Shlomi Dolev, Elad Schiller, and Jennifer Welch. Random walk for self-stabilizing group communication in ad-hoc networks. *IEEE Transactions on Mobile Computing*, 5(7):893–905, 2006.

[26] Peter Druschel and Antony Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HOTOS'01)*, pages 75–80. IEEE, 2001.

[27] Website of eDonkey. http://www.edonkey2000.com.

[28] Paul Erdős and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae, Debrecen*, 6:290–297, 1959.

[29] Uriel Feige. A tight lower bound on the cover time for random walks on graphs. *Random Structures & Algorithms*, 6(4):433–438, 1995.

[30] Niels Ferguson and Bruce Schneier. *Practical cryptography*, volume 141. Wiley New York, 2003.

[31] Santo Fortunato and Alessandro Flammini. Random walks on directed networks: the case of PageRank. *International Journal of Bifurcation and Chaos*, 17(07):2343–2353, 2007.

[32] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.

[33] Website of Gnutella. http://www.gnutella.com.

[34] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in dynamic structured P2P systems. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, volume 4, pages 2253–2262. IEEE, 2004.

[35] Krishna P Gummadi, Richard J Dunn, Stefan Saroiu, Steven D Gribble, Henry M Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 314–329. ACM, 2003.

[36] Nicholas JA Harvey, Michael B Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. SkipNet: A scalable overlay network with practical locality properties. In *USENIX Symposium on Internet Technologies and Systems*, volume 274, 2003.

[37] Carlos P Herrero. Self-avoiding walks on scale-free networks. *Physical Review E*, 71(1):016103, 2005.

[38] Phouvieng Hieungmany and Shigeo Shioda. Characteristics of random walk search on embedded tree structure for unstructured P2Ps. In *Proceedings of the 16th International Conference on Parallel and Distributed Systems (ICPADS'10)*, pages 782–787. IEEE, 2010.

[39] Barry D Hughes. Random walks and random environments: Volume 1: Random walks. 1996.

[40] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nick Briggs, and Rebecca Braynard. Networking named content. *Communications of the ACM*, 55(1):117–124, 2012.

[41] Mihajlo A Jovanovic, Fred S Annexstein, and Kenneth A Berman. Scalability issues in large peer-to-peer networks - a case study of Gnutella. Technical report, Technical report, University of Cincinnati, 2001.

[42] Jeff D Kahn, Nathan Linial, Noam Nisan, and Michael E Saks. On the cover time of random walks on graphs. *Journal of Theoretical Probability*, 2(1):121–128, 1989.

[43] Vana Kalogeraki, Dimitrios Gunopulos, and Demetrios Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *Proceedings of the 11th International Conference on Information and Knowledge Management (2002)*, pages 300–307. ACM, 2002.

[44] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997)*, pages 654–663. ACM, 1997.

[45] Website of Kazaa. http://www.kazaa.com.

[46] Balachander Krishnamurthy, Jia Wang, and Yinglian Xie. Early measurements of a cluster-based architecture for P2P systems. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (2001)*, pages 105–109. ACM, 2001.

[47] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM Sigplan Notices*, 35(11):190–201, 2000.

[48] Thomas Kuczek and Keith Crank. On a self-avoiding random walk. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 54–66, 1994.

[49] Yoram Kulbak, Danny Bickson, et al. The eMule protocol specification. *eMule project, http://sourceforge.net*, 2005.

[50] Ching Law and K-Y Siu. Distributed construction of random expander networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications IEEE Societies (INFOCOM 2003)*, volume 3, pages 2133–2143. IEEE, 2003.

[51] Sang Hoon Lee, Pan-Jun Kim, and Hawoong Jeong. Statistical properties of sampled networks. *Physical Review E*, 73(1):016102, 2006.

[52] Víctor M López Millán, Vicent Cholvi, Luis López, and Antonio Fernández Anta. A model of self-avoiding random walks for searching complex networks. *Networks*, 60(2):71–85, 2012.

[53] Víctor M. López Millán, Vicent Cholvi, Luis López, and Antonio Fernández Anta. Resource location based on partial random walks in networks with resource dynamics. In *Proceedings of the 4th International Workshop on Theoretical Aspects of Dynamic Distributed Systems (TADDS'12)*, pages 26–31. ACM, 2012.

[54] Víctor M. López Millán, Vicent Cholvi, Luis López, and Antonio Fernández Anta. Improving resource location with locally precomputed partial random walks. In *LNCS, Proceedings of 1st the International Conference on Networked Systems (NETYS'13)*, volume 7853, pages 144–158. Springer-Verlag, 2013.

[55] László Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdős is eighty*, 2(1):1–46, 1993.

[56] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing (2002)*, pages 84–95. ACM, 2002.

[57] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. Can heterogeneity make Gnutella scalable? In *Peer-to-Peer Systems*, pages 94–103. Springer, 2002.

[58] Issam Mabrouki, Xavier Lagrange, and Gwillerm Froc. Random walk based routing protocol for wireless sensor networks. In *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools (2007)*, page 71. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[59] Neal Noah Madras and Gordon Slade. *The self-avoiding walk*. Birkhäuser, 2013.

[60] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC'02)*, pages 183–192. ACM, 2002.

[61] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: The power of lookahead in randomized P2P networks. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (2004)*, pages 54–63. ACM, 2004.

[62] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[63] Daniel A Menascé and Lavanya Kanchanapalli. Probabilistic scalable P2P resource location services. *ACM SIGMETRICS Performance Evaluation Review*, 30(2):48–58, 2002.

[64] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*, chapter 6, pages 127–160. Cambridge University Press, 1995.

[65] Website of Napster. http://www.napster.com.

[66] Mark E J Newman, Albert-László Barabási, and Duncan J Watts. *The structure and dynamics of networks*. Princeton University Press, 2006.

[67] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118, 2001.

[68] Website of Overnet. http://www.overnet.com.

[69] Sagar A Pandit and R E Amritkar. Random spread on the family of small-world networks. *Physical Review E*, 63(4):041104, 2001.

[70] C Greg Plaxton, Rajmohan Rajaraman, and Andrea W Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(3):241–280, 1999.

[71] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. *A scalable content-addressable network*, volume 31. ACM, 2001.

[72] Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword searching. In *LNCS, Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware (2003)*, pages 21–40. Springer-Verlag New York, Inc., 2003.

[73] Sean C Rhea and John Kubiatowicz. Probabilistic location and routing. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, volume 3, pages 1248–1257. IEEE, 2002.

[74] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (2001)*, pages 99–100. IEEE, 2001.

[75] Jordan Ritter. Why Gnutella can't scale. No, really, 2001.

[76] Luis Rodero Merino. *Self-Adaptation Mechanisms for Efficient Resource Location in Peer-to-Peer Systems*. PhD thesis, Universidad Rey Juan Carlos, 2007.

[77] Luis Rodero-Merino, Antonio Fernández Anta, Luis López, and Vicent Cholvi. Self-managed topologies in P2P networks. *Computer Networks*, 53(10):1722–1736, 2009.

[78] Luis Rodero-Merino, Antonio Fernández Anta, Luis López, and Vicent Cholvi. Performance of random walks in one-hop replication networks. *Computer Networks*, 54(5):781–796, 2010.

[79] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*, pages 329–350. Springer, 2001.

[80] Narayanan Sadagopan, Bhaskar Krishnamachari, and Ahmed Helmy. Active query forwarding in sensor networks. *Ad Hoc Networks*, 3(1):91–113, 2005.

[81] S. Saroiu, P. K. Gummadi, and S. D. Gribble. Measurement study of peer-to-peer file sharing systems. In *Proceedings of SPIE, Multimedia Computing and Networking (MMCN'02)*, volume 4673, pages 156–170, 2001.

[82] Nima Sarshar and Vwani Roychowdhury. Scale-free and stable structures in complex ad hoc networks. *Physical Review E*, 69(2):026101, 2004.

[83] Gordon Slade. The diffusion of self-avoiding random walk in high dimensions. *Communications in Mathematical Physics*, 110(4):661–683, 1987.

[84] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 149–160. ACM, 2001.

[85] Bosiljka Tadić. Adaptive random walks on the class of Web graphs. *The European Physical Journal B*, 23(2):221–228, 2001.

[86] Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P'03)*, pages 102–109. IEEE, 2003.

[87] Dimitrios Tsoumakos and Nick Roussopoulos. Analysis and comparison of P2P search methods. In *Proceedings of the 1st International Conference on Scalable Information Systems (2006)*, page 25. ACM, 2006.

[88] Sameer Tuladhar and L. Sumalatha. Replication of query messages in the unstructured overlays peer-to-peer network. *International Journal of Engineering Research and Applications (IJERA)*, 2:1063–1066, 2012.

[89] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of the 22th International Conference on Distributed Computing Systems (2002)*, pages 5–14. IEEE, 2002.

[90] Shi-Jie Yang. Exploring complex networks by walking on them. *Physical Review E*, 71(1):016107, 2005.

[91] Ben Yanbin Zhao, John Kubiatowicz, Anthony D Joseph, et al. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California, Berkeley, 2001.

[92] David Zuckerman. A technique for lower bounding the cover time. *SIAM Journal on Discrete Mathematics*, 5(1):81–87, 1992.

Appendix A

Glossary of Terms

Attachment point: A point in a node where one of the two ends of a link is attached (see *graph*).

Autolink: A link whose two endpoints are connected to the same node (see *simple graph*).

Average search length: See *search length*.

Bloom filter: A probabilistic data structure that efficiently stores a set of elements in a fixed space. Bloom filters support membership queries. Efficiency in the space usage comes at the cost of introducing *false positives*. A false positive occurs when the filter is queried about a given element not stored in the filter (not belonging to the set) and the returned result is positive. False positives happen with a probability p that depends on the number of elements stored in the filter and on the storage space.

Check-first/Choose-first: In the resource location mechanisms based on partial walks (PW-RW and PW-SAW), they are two alternative procedures to choose a partial walk among the ones available at a given node, when a search needs to decide whether to jump over a partial walk or to traverse it looking for the desired resource. *Choose-first* consists of choosing one partial walk uniformly at random among all partial walks in the node, and then checking (querying) it for the resource. *Check-first*, on the other hand, first checks all partial walks in the node for the resource, and then chooses uniformly at random among those that returned a positive result, if any, or otherwise among all partial walks in the node. These two variations result in four mechanisms based on partial walks: choose-first PW-RW and PW-SAW, and check-first PW-RW and PW-SAW.

Connected graph: A graph in which any vertex can be reached from any other vertex following the edges of the graph.

Covered node: A node whose resources a walker (search) knows about. That is, the walker has either visited the node itself or one of its neighbors in a network with one-hop resource replication. The number of nodes covered by a SAW is denoted by C.

Degree: The *degree* of a vertex or node of an undirected and simple graph or network is the number of edges that connect that node to others. It is denoted by k. According to this definition, a node has a number of neighbors equal to its degree.

Degree distribution: The degree distribution of a graph or network is the probability distribution of the degrees over all the nodes of the graph/network. It is denoted by p_k.

Distributed Hash Table (DHT): Distributed system providing a resource storage and look-up service similar to a hash table. The network store pairs (*key*, *value*) and the mapping among keys and nodes is distributed among the nodes. Any node can efficiently retrieve the value associated to a key.

Edge: An edge in a graph $G(V, E)$ is any of the elements of E (see *graph*).

Endpoint: An endpoint of an edge in a network is each of the nodes connected by that edge.

Erdős-Rényi network: A random network that is constructed establishing a link between any pair of distinct nodes with a certain probability characteristic of the network.

Expected degree: The expected degree of a graph or network is the expectation of the degree of its nodes. It is denoted by $\overline{k}$.

Expected search length: See *search length*.

False positive: A possible result of a query to a Bloom filter (see *Bloom filter*). False positives can also result from queries when deterministic data structure are used in dynamic resources scenarios. In this case, false positives are due to information deterioration over time.

Finite graph: A graph in which the set of vertices has a finite size $|V|$.

Graph: A graph is a representation of a set of objects some of which are connected with each other. We use this concept to represent any kind of communication network consisting of nodes connected by links. The terms *graph* and *network* are used equivalently throughout this dissertation.

In graph theory, a *graph* G is defined as an unordered pair $G(V, E)$ where V is a set of *vertices* or *nodes* and E is a set of *edges*, *arcs*, *lines* or *links*. An edge is a 2-element subset of V.

Hop: The movement taken by a walker from the current node to one of its neighbors in any kind of random walk. In the context of resource location, hops often follow the edges in an overlay network. In this case, one *hop* of the walker generally implies the traversal of several links in the underlying network.

In the resource location mechanisms based on partial walks proposed in this dissertation, hops can be either *jumps* or *steps*, which in turn can be *unnecessary steps* or *trailing steps*.

Instance: See *resource multiplicity*.

Jump: In the resource location mechanisms based on partial walks proposed in this dissertation, a *jump* is the movement taken by a walker from the first node of a partial walk to the last node of that partial walk (using the underlying network), when the desired resource is known not to be in any of its nodes. The expected number of jumps in searches is denoted by $\overline{J}$.

k-random walk: When a resource location mechanism launches k walkers (random walks) from the target node simultaneously to find the desired resource, it is said that it uses a *k-random walk*.

k-regular network/graph: A network in which all nodes/vertices have degree k. k is referred to as the *degree of the network*. The topology of k-regular networks is not restricted in any way except in that all nodes need to have exactly k neighbors.

Link: A link in a network is any of the connections between the nodes of the network. In the context of this dissertation, a *link* is equivalent to a *edge* in a graph.

Loop: A loop in a graph is an edge whose two endpoints are the same vertex. Not to be confused with a *cycle*, which is a path (sequence of

adjacent edges) that, starting from a given vertex, allows to reach the same vertex again (see *simple graph*).

Multilink: A multilink in a network is a set of more than one link (or edge) connecting the same pair of nodes (see *simple graph*).

Neighbor: A neighbor of a node i in a graph or network is another node connected to i by an edge or link of the graph. Neighbors in an overlay network are not in general neighbors in the underlying network.

Network size: The number of nodes of the network, denoted as N. In graph theory, $|V|$ is called the graph's *order*, while $|E|$ is called the graph's *size*. In this dissertation, we use the term *network size* for $|V|$, as is usual in the networks community.

Node: A node of a network is any of its active elements, possibly connected to others by links. In the context of this dissertation, a *node* is equivalent to a *vertex* in a graph. In particular, since the network is typically an overlay, the nodes of a network are end systems connected to other end systems by means of the underlying network.

One-hop replication: See *resource replication*.

Optimal partial walk length: See *partial walk length*.

Optimal search length: See *partial walk length*.

Overlay network: A network whose nodes are some subset of the end systems connected to a physical network (the *underlying* network), and whose links are logical relations established among them typically according to some service logic.

Partial walk: A partial random walk, or partial walk for short (PW), is a random walk of some fixed length which is used, connecting it to other partial walks, to construct a longer random walk. Partial walks are used in some of the resource location mechanisms proposed in this dissertation. The number of partial walks precomputed by each node of the network is denoted by w.

Partial walk length: The number of hops that a random walk takes to compute the partial walk, denoted by s. It is a fixed parameter in the proposed mechanisms based on partial walks. Note that the number of nodes in the partial walk, including the first node (the one that computes the partial walk) and the last node, is $s + 1$. When a given

partial walk length produces the minimum expected search length, it is called the *optimal partial walk length*, and it is denoted by s_{opt}. The corresponding expected search length is called the *optimal search length* and is denoted as $\overline{L}_{opt}$.

PW-RW/PW-SAW: Resource location mechanisms proposed in this dissertation, based on partial walks that are normal random walks and self-avoiding random walks, respectively.

Random graph: A random graph is a graph constructed by a random process. Sometimes, the term refers to an Erdős-Rényinetwork, a particular type of random graph.

Random walk: A random walk (RW) in a network is a routing mechanism by which the next node to visit is chosen uniformly at random among the neighbors of the current node. This concept is central in this dissertation, since all resource location mechanisms proposed and studied are based on random walks. When we want to emphasize the distinction between RWs and other types of walks we refer to the latter as *simple random walks*.

Resource: A resource is any object (file, service, attribute, etc.) held by a node in a network that another node desires to locate.

Resource multiplicity: Resource multiplicity in a network whose nodes hold resources is the property by which there can be multiple *instances* of a each resource held by different nodes of the network. The number of instances of a given resource is denoted by R.

Resource replication: Resource replication in a network whose nodes hold resources is the property by which a node knows about the resources held by its neighbors and its neighbor's neighbors up to a given number of hops. In this dissertation we consider *one-hop replication* networks as a network model in which each node knows about the resources held by its neighbors.

Scale-free network: A scale-free network is a network with a degree distribution that follows a power law, at least for large values of the degree k: $P(k) \propto k^{-\gamma}$. In real networks observed to have this type of degree distribution, their exponent are such that $2 < \gamma < 3$.

Search: A search in a network is a walker that aims to visit a node that holds some resource, starting from a node that desires to locate that resource.

Search length: When locating a resource in a network, the search length is the number of hops taken to find the desired resource. The *expected search length* is the expectation of the random variable representing search lengths in a given analytical model. The *average search length* is the statistical mean of the lengths of the searches performed in an experiment. Expected search lengths are denoted using upper case, while average search length are denoted using lower case. For example, $\overline{L}_{rw}$, $\overline{L}_{saw}$, and $\overline{L}_s$ are the expected search lengths achieved by simple RW, SAW, and PW of length s, respectively. The first one is also denoted by $\overline{L}$.

Self-avoiding random walk: A self-avoiding random walk (SAW) is a random walk that tries to avoid revisiting nodes, i.e., that tries not to visit nodes that have been already visited by the walk. Several variations of SAW are defined in the literature. The SAW used in some of the proposed resource location mechanisms simply chooses uniformly at random among the unvisited neighbors of the current node. If all nodes have been already visited, the walk behaves as a simple random walk.

Simple graph: A graph is simple when loops and multilinks are not allowed. Graphs considered in this dissertation are simple because they properly model most communication networks.

Step: In the resource location mechanisms based on partial walks proposed in this dissertation, a *step* is each of the hops taken by a walker when traversing a partial walk, i.e., when proceeding through it node by node looking for the resource. If the decision of traversing the partial walk was based on false information, the resource will not be found. Then, the hops taken to traverse the entire partial walk are called *unnecessary steps*. Otherwise, the hops taken until the resource is found are called *trailing steps*. The expected number of unnecessary and trailing steps in searches are denoted by $\overline{U}$ and $\overline{T}$, respectively.

Topology: The topology of a graph or network is the way that vertices or nodes are connected by edges or links. The topology gives information only about what nodes are connected to each node, excluding other attributes of the links themselves (type, length, bandwidth, cost, etc.).

Trailing step: See *step*.

TTL: Time To Live refers to any mechanism that limits the length of a random walk if it does not locate the desired resource. The walker is

then stopped (the search message is discarded) and the searches finishes unsuccessfully. A similar mechanism is also usually applied to flooding when it is used for searching in unstructured decentralized systems.

Underlying network: A generally physical network that supports a logical network among a subset of the end systems connected to it. The logical network is referred to as the overlay network. Overlay networks are used as the basis of some resource location mechanisms.

Undirected graph: A graph in which edges have no orientation (ordering between its connected vertices). Graphs considered in this dissertation are undirected since they properly model most communication networks.

Unnecessary step: See *step*.

Vertex: A vertex in a graph $G(V, E)$ is any of the elements of V (see *graph*).

Visited node: A node which has been visited by a walker (search), i.e., the search has arrived to it in its quest for the desired resource, and then it has left the node for one of its neighbors if the resource was not found. The number of nodes visited by a SAW is denoted by V.

Walk: A walk in a network is a sequence of nodes where each node is connected to the previous one by a link of the network.

Walker: A particular state of a search for a desired resource in which it is being processed by a given node at a given time.

Appendix B

Resumen en español

Este apéndice presenta un resumen extendido en español del contenido de esta tesis doctoral.[1]

B.1. Antecedentes

El trabajo de investigación presentado en esta tesis se enmarca en el campo de los sistemas distribuidos, y en particular, en el área de la búsqueda de recursos en redes complejas. Puede definirse el problema de la *búsqueda* o *localización de recursos* como encontrar un nodo de la red que tenga un recurso en particular (un fichero, un servicio, un dispositivo, un identificador, etc.). Este problema surge en multitud de redes reales, como por ejemplo los *sistemas entre pares* (*peer-to-peer* or P2P systems), redes inalámbricas ad-hoc, redes de contenidos, etc.

Los mecanismos de búsqueda utilizan algoritmos para navegar por la red mediante mensajes partiendo del nodo que quiere encontrar el recurso. Estos algoritmos utilizan como entrada información sobre la red para guiar la búsqueda. En general, cuanto mayor es esta información mejor será el rendimiento de la búsqueda. El disponer de dicha información conlleva, no obstante, el coste correspondiente a obtenerla y gestionarla. El rendimiento puede medirse mediante magnitudes como el tiempo que dura la búsqueda, la longitud de la búsqueda (número de saltos para encontrar el recurso), o el ancho de banda utilizado por los mensajes que se envían. Los mecanismos de búsqueda de tipo centralizado utilizan un directorio central que registra los recursos y su localización en la red. Los mecanismos distribuidos evitan el servidor central con diferentes estrategias, como por ejemplo imponer una cierta

[1] En cumplimiento de los requisitos establecidos en el Artículo 24 de la Normativa Reguladora de Tercer Ciclo de la Universidad Rey Juan Carlos.

estructura en la topología de la red que facilite la localización de los recursos. Dicha red será seguramente una *red superpuesta* (*overlay network*), formada mediante el establecimiento de enlaces lógicos entre los distintos extremos de la red. Algunos algoritmos de búsqueda son determinísticos, mientras que otros son aleatorios.

El objetivo general de esta tesis es diseñar mecanismos eficientes de búsqueda de recursos en una red con la menor información de entrada posible, así como evaluar su rendimiento utilizando tanto modelos analíticos como experimentos mediante simulaciones. En particular, esta tesis propone la utilización de *caminos* o *paseos aleatorios* (*random walks*), que han demostrado ser útiles en redes de las que se conoce poca o ninguna información, frecuentemente debido a su gran tamaño, a su carencia de topologías o mecanismos de construcción definidos, y a que pueden cambian dinámicamente con el tiempo. Dichas redes se conocen *redes complejas*, y responden a las características de muchas redes reales (Internet, WWW, P2P, wireless ad-hoc networks, etc.). Dichas redes exhiben características particulares que no surgen en el estudio tradicional de la teoría de grafos. Los caminos aleatorios pueden utilizarse para buscar recursos en una red. El único requisito de información de entrada es que cada nodo conozca sus vecinos, para poder elegir de forma aleatoria con una distribución uniforme el siguiente nodo a visitar si el nodo actual no posee el recurso buscado. Sin embargo, la longitud media de las búsquedas por caminos aleatorios resulta ser elevada.

Definimos un *camino aleatorio* en una red como un mecanismo de encaminamiento que elige el siguiente nodo a visitar de forma uniformemente aleatoria entre los vecinos de nodo actual. Los caminos aleatorios han sido estudiados extensamente en matemáticas, donde se han analizado como cadenas de Markov finitas [39, 55, 66], y se han utilizado en una amplia variedad de campos tales como la física estadística, la dinámica de poblaciones, la bioinformática, etc. Aplicados a las redes de comunicaciones, los caminos aleatorios han tenido un profundo impacto en la algoritmia y la teoría de la complejidad. Algunas de sus ventajas son la simplicidad, el bajo consumo de recursos de procesamiento y almacenamiento en los nodos, y el hecho de que utilizan sólo información local, evitando la sobrecarga de comunicaciones necesaria en otros mecanismos de encaminamiento. Los caminos aleatorios han sido propuestos como mecanismos básicos para multitud de aplicaciones, incluyendo el muestreo, la búsqueda, la construcción y la caracterización de redes [2, 3, 10, 16, 25, 31, 32, 50, 51, 57, 56, 58, 78, 80, 85]. También se han propuesto los caminos aleatorios como una posible solución al problema de búsqueda de recursos. El mecanismo de búsqueda es el siguiente. El nodo origen de la búsqueda comprueba si él mismo tiene el recurso. En caso contrario, elige un vecino aleatoriamente de forma uniforme y le envía un

mensaje indicando el recurso deseado. En otras palabras, la búsqueda *salta* a dicho vecino. Éste actúa de la misma manera, comprobando si tiene el recurso y reenviando el mensaje a uno de sus vecinos en caso contrario. La búsqueda termina cuando se encuentra el recurso, momento en el que el nodo que lo posee envía un mensaje al nodo original para comunicárselo. Debido al carácter aleatorio del camino, algunos nodos serán visitados más de una vez (innecesariamente, desde el punto de vista de la búsqueda), mientras que otros nodes permanecerán sin visitar durante largo tiempo. El número de saltos necesarios para encontrar el recurso se denomina *longitud de la búsqueda*. El rendimiento de esta aplicación directa de los caminos aleatorios se ha estudiado en [2, 32, 56, 78, 90].

La utilización de caminos aleatorios en la búsqueda de recursos tiene varias posibles aplicaciones, como por ejemplo las *redes centradas en contenidos* (*content-centric networks* or CCN) o los sistemas de compartición de ficheros P2P no estructurados. Las primeras son redes en las que los elementos clave son porciones de contenidos que son solicitadas por los usuarios indicando el nombre correspondiente. Dichas porciones de contenido tienen que localizarse y transferirse de forma eficiente para su consumo por los usuarios. La aplicación de los caminos aleatorios a los sistemas del último tipo se describe extensamente en el Capítulo 2.

B.2. Objetivos

Como se ha mencionado en el apartado anterior, el objetivo de la investigación presentada es el diseño, modelado y evaluación de mecanismos de búsqueda de recursos eficientes basados en caminos aleatorios. En concreto, se formula la siguiente hipótesis dentro del correspondiente contexto:

Contexto: Consideramos redes construidas de forma aleatoria cuyos nodos poseen recursos. Los nodos no tienen información sobre la topología de la red salvo el conocimiento de los nodos directamente conectados a ellos. Los nodos están interesados en encontrar recursos que se encuentran en uno o más nodos de la red. Las búsquedas de dichos recursos pueden ser realizadas utilizando caminos aleatorios normales.

Hipótesis: Pueden utilizarse variaciones de los caminos aleatorios o combinarse éstos con otros mecanismos para diseñar algoritmos de localización de recursos en redes cuyo rendimiento supera al de las búsquedas basadas en caminos aleatorios normales en cuanto a la longitud media de la búsqueda.

A continuación se desarrolla el objetivo general anterior en objetivos concretos. Las ideas principales utilizadas en los mecanismos de búsqueda propuestos son:

- La utilización de caminos aleatorios de tipo SAW (*self-avoiding walks*) en lugar de caminos aleatorios normales.
- El uso de caminos aleatorios parciales precomputados para construir caminos aleatorios más largos de forma eficiente.
- La combinación de caminos parciales y caminos aleatorios SAW.
- El uso de almacenamiento de información determinístico y probabilístico en las búsquedas.
- La definición de variantes de los mecanismos de búsqueda propuestos, derivadas de sus características particulares.

La evaluación del rendimiento de los mecanismos de búsqueda propuestos ha sido realizada en diferentes tipos de red y de asignación de recursos a nodos:

- Varios tamaños de red y distribuciones de grado de los nodos.
- La disponibilidad o no de replicación de recursos en la red.
- La existencia de una o varias instancias del recurso buscado en la red.
- El comportamiento estático o dinámico de los recursos de la red.
- Varias parametrizaciones de cada uno de los mecanismos, de acuerdo con sus características particulares.

Los siguientes apartados detallan los mecanismos de búsqueda propuestos:

B.2.1. Caminos aleatorios SAW

Una posibilidad para mejorar el rendimiento de los caminos aleatorios normales es modificarlo ligeramente para intentar no revisitar nodos ya visitados por ese camino, incrementando el número de nodos nuevos visitados y elevando por lo tanto la probabilidad de visitar el nodo que posee el recurso deseado. Dichos caminos aleatorios reciben el nombre de *self-avoiding random walks* o SAW. La definición de este tipo de caminos aleatorios en matemáticas ha sido adaptada a nuestro objetivo de búsqueda de recursos en redes de comunicaciones.

El objetivo principal en relación a SAW es la elaboración de un modelo analítico que estime de forma precisa el rendimiento medio de las búsquedas de este mecanismo en redes con o sin replicación de recursos y en las que pueda haber múltiples instancias del recurso buscado. En las redes con replicación de recursos, los nodos tienen conocimiento de los recursos que poseen sus vecinos además de los recursos propios. Este trabajo considera replicación de recursos *a un salto* (*one-hop replication*), es decir, los nodos conocen sus recursos y los de sus vecinos directos. Por lo tanto, para encontrar un recurso basta con visitar el nodo que lo posee o cualquiera de sus recursos. Esta característica, que contribuye a disminuir la longitud de las búsquedas con un coste limitado, ha sido incluida como parte de otros mecanismos de búsqueda en sistemas P2P [16, 18, 61, 78].

Se pretende evaluar el rendimiento del mecanismo propuesto haciendo uso tanto del modelo analítico desarrollado como de experimentos con simulaciones, que permiten validar las prediciones del modelo. Se desea comparar dicho rendimiento con el de las búsquedas basadas en caminos aleatorios normales. El estudio de prestaciones se plantea para tres tipos de redes y con varios juegos de parámetros de entrada, al igual que para el resto de los mecanismos.

B.2.2. Caminos aleatorios parciales

Otra posibilidad para mejorar el rendimiento de las búsquedas con caminos aleatorios consiste en modificar el procedimiento de construcción del camino aleatorio normal. La idea consiste en construir un camino aleatorio conectando entre sí caminos aleatorios parciales que están disponibles en cada nodo, de forma mucho más rápida que la construcción normal (secuencial) de dicho camino. Los caminos parciales, que tienen una longitud fija y relativamente corta, son precomputados por cada nodo en una fase inicial anterior a las búsquedas. Cuando se construye cada camino parcial, se registran los recursos que poseen los nodos que forman parte del camino. Esta información será luego utilizada por las búsquedas. Se propone la utilización de estructuras de datos probabilísticas, necesarias en el caso de que el volumen de información sea elevado. Sin embargo, dichas estructuras probabilísticas, en concreto filtros de Bloom [12, 14], introducen *falsos positivos* al interrogar un PW por el recurso deseado.

Las búsquedas se realizan de acuerdo al siguiente mecanismo. El nodo origen de la búsqueda elige un camino parcial de los disponibles en dicho nodo. Después, se comprueba si el recurso buscado está en dicho camino, interrogando su filtro de Bloom. En caso positivo, se recorre el camino nodo a nodo hasta visitar el nodo que contiene el recurso, momento en el que ter-

mina la búsqueda. En caso negativo, se *salta* al último nodo de ese camino parcial (ahorrando los saltos correspondientes a recorrer el camino entero). En dicho nodo se repite el proceso, conectando al anterior uno de sus caminos parciales y procediendo de la misma manera que el nodo origen. Al interrogar un filtro de Bloom puede obtenerse un falso positivo, cuyo resultado es que la búsqueda recorre el camino parcial paso a paso sin encontrar el recurso. Cuando se alcanza el último nodo, se elige un nuevo camino aleatorio, procediéndose como se ha descrito. Un variante de este mecanismo consiste en interrogar todos los caminos parciales del nodo para ver si alguno tiene el recurso, eligiéndose en ese caso entre los que den un resultado positivo.

Los objetivos principales con respecto a los caminos aleatorios parciales son el desarrollo de un modelo analítico para la estimación de la longitud media de las búsquedas, y la realización del correspondiente evaluación de su rendimiento de forma analítica y experimental. Además, se desea saber cuál es la longitud óptima de los caminos parciales, el efecto del número de caminos parciales precomputados por cada nodo, y cuál es el impacto de los falsos positivos. También se pretende comparar el rendimiento de las dos variantes descritas, así como el efecto de utilizar caminos aleatorios normales o SAW como caminos parciales.

B.2.3. Caminos parciales con recursos dinámicos

Como se ha descrito en el apartado anterior, los caminos aleatorios suponen la recopilación de información sobre recursos en la red *anteriormente* a la realización de las búsquedas. Si la asignación de recursos a nodos varía con el tiempo, esto supone que la información recopilada sufre un progresivo deterioro, que afectará al rendimiento de los mecanimsmos basados en caminos aleatorios. Por ejemplo, supongamos que al interrogar el filtro de Bloom de un camino parcial hemos obtenido una respuesta positiva. Si el recurso ha desaparecido después del nodo en cuestión, la búsqueda no lo encontrará al recorrer el camino parcial. Con el objetivo de medir el impacto de este efecto, se plantea el estudio de dichos mecanismos en redes con recursos dinámicos (es decir, que pueden aparecer y desaparecer de los nodos). Para aislar el efecto del deterioro de la información, se propone en este modelo el uso de estructuras de datos tradicionales (determinísticas).

Como en los casos anteriores, los objetivos comprenden el desarrollo de los modelos analíticos necesarios para adaptar los mecanismos anteriores al nuevo contexto de recursos dinámicos. El estudio de rendimiento pretende comparar las búsquedas basadas en caminos parciales en redes con recursos estáticos y dinámicos, y también con las redes basadas en caminos aleatorios normales.

B.3. Metodología

El trabajo de investigación de esta tesis se ha realizado de acuerdo con la siguiente metodología. Para cada uno de los mecanismos de búsqueda propuestos:

1. Se concibe una idea para mejorar el rendimiento de las búsquedas por caminos aleatorios normales.

2. Se desarrolla un modelo que describe el mecanismo de búsqueda que plasma la idea anterior. En este modelo se definen las distintas variantes del mecanismo, en su caso.

3. Se desarrolla un modelo analítico para la obtención de las distintas magnitudes del modelo anterior, que nos permiten llegar a expresiones para la esperanza de las longitudes de las búsquedas, entre otras.

4. Se realiza una herramienta para el cálculo del modelo a partir de los parámetros de entrada.

5. Se modifica el simulador de eventos discretos desarrollado expresamente para incorporar el modelo definido, con los parámetros correspondientes, de forma que se integre con el resto de las funcionalidades ya existentes.

6. Se definen experimentos como conjuntos de parámetros de entrada.

7. Se alimentan tanto el simulador como la herramienta de cálculo del modelo analítico con los parámetros de entrada anteriores para cada experimento. Los resultados analíticos y de simulación se comparan para validar el modelo y evaluar la precisión de sus resultados.

8. Se extraen conclusiones sobre el rendimiento del mecanismo de búsqueda de los resultados anteriores. En particular, se compara el rendimiento del mecanismo con las búsquedas basadas en caminos aleatorios normales.

A continuación se presentan el marco metodológico común a los mecanismos propuestos.

B.3.1. Modelos de red

Se consideran *redes aleatorias*, es decir, construidas mediante un proceso aleatorio. Además, se requiere que la red sea *aleatoria* en otro sentido: que cualquier punto de conexión de un nodo dado (donde se conecta un extremo de un enlace) pueda estar conectado a cualquier otro punto de conexión en la red. El proceso de construcción de red empleado [67] garantiza esta característica. Para la construcción de las redes se parte de un tamaño de red (N) y de una distribución de grado concreta (p_k), de acuerdo con su tipo. Esta información se utiliza como entrada para los modelos analíticos y los experimentos de simulación, pero no es necesaria para los mecanismos de búsqueda en sí. Los experimentos se realizan para tres tipos de redes:

- **Redes regulares de grado** k (*k-regular networks*), cuyos nodos tienen todos grado k, siendo los enlaces establecidos aleatoriamente entre los nodos.
- **Redes Erdős-Rényi** (ER), caracterizadas porque el enlace entre cada dos nodos de la red se establece con una cierta probabilidad fija al construir la red.
- **Redes libres de escalas o polinómicas** (*scale-free* o *power-law*), cuya distribución de grado sigue una ley polinómica. Su interés radica en que algunas redes reales (WWW, redes sociales, etc.) siguen aproximadamente una distribución de este tipo.

B.3.2. Modelos de búsquedas

Una búsqueda se define como un camino (*walk*) en la red cuyo objetivo es encontrar un determinado recurso. Este recurso puede ser único en la red (reside en un solo nodo) o puede haber varias instancias del recurso en múltiples nodos. En los modelos analíticos y en los experimentos se garantiza la existencia de al menos una instancia del recurso buscado. El nodo que lanza una búsqueda para encontrar un cierto recurso se denomina *origen* (*source node*). El *caminante* (*walker*) se desplaza de un nodo a otro (mediante mensajes) hasta que visita un nodo *destino*, que posee el recurso buscado (*target node*). De forma alternativa, si cada nodo tiene conocimiento de los recursos albergados por sus vecinos, bastará con que el caminante visite alguno de los vecinos de un nodo destino. Esta característica, que disminuye las longitudes de las búsquedas, se denomina *replicación de recursos a un salto* (*one-hop replication*).

La principal magnitud que deseamos estimar y medir es la *longitud media de la búsqueda*. La longitud de la búsqueda se define como el número

de saltos que un caminante tiene que hacer para localizar el recurso. Los modelos analíticos calculan la *esperanza matemática* de las longitudes de las búsquedas, mientras que en los experimentos de simulación se calcula la *media estadística* de las longitudes de las búsquedas realizadas. Para cada una de las búsquedas simuladas, tanto el nodo origen como el destino se eligen aleatoriamente con distribución uniforme en los nodos de la red. El objetivo es contar con un modelo analítico que nos permita predecir la longitud media de las búsquedas mediante su esperanza de forma lo más precisa posible.

B.4. Conclusiones

Los modelos analíticos desarrollados para las búsquedas con caminos aleatorios SAW y con caminos aleatorios parciales (tanto en redes con recursos estáticos o dinámicos) pueden considerarse como una de las contribuciones más importantes de la investigación realizada. Además, el estudio de prestaciones de cada mecanismo ha permitido extraer conclusiones que se resumen a continuación. Los SAW consiguen reducciones de la longitud media de las búsquedas con respecto a los caminos aleatorios normales de entre el 50 % y el 75 % en los experimentos realizados, dependiendo del tipo de red y de su grado medio. Las redes *scale-free* son las que exhiben reducciones mayores, que aumentan al incrementar el grado medio de la red (observándose el efecto inverso en las redes regulares y ER). En redes con replicación de recursos, las reducciones disminuyen a entre el 7 % y el 26 %, mostrando que los beneficios de intentar no revisitar nodos se ven compensados parcialmente por el conocimiento de los recursos de los vecinos cuando se usan caminos aleatorios normales.

Las búsquedas con caminos aleatorios parciales consiguen reducciones mayores de sus longitudes. En particular, esta reducción es del orden de la raíz cuadrada de la longitud media de las búsquedas de los caminos aleatorios normales, cuando la probabilidad de falso positivo en los filtros de Bloom es pequeña. Esto supone unas reducciones de entre el 88 % y el 98 % en los experimentos realizados, sin una dependencia importante con el tipo de red. Al combinar los caminos parciales con los caminos SAW se obtienen reducciones adicionales de entre un 5 % y un 12 %. Una variante de los caminos parciales consigue reducciones superiores al incrementar el número de caminos parciales precalculados por cada nodo, a expensas de incrementar el coste de la fase de construcción de éstos.

Cuando se adaptan los mecanismos basados en caminos aleatorios parciales a redes con recursos dinámicos se observan también importantes reducciones de la longitud media de las búsquedas. Los experimentos realizados

han mostrado que estas reducciones disminuyen al aumentar la dinamicidad de dichos recursos, pero se mantienen en valores significativos incluso para una alta volatilidad.

Todo ello permite concluir que la hipótesis planteada queda demostrada convirtiéndose en tesis:

> **Tesis:** Pueden utilizarse variaciones de los caminos aleatorios o combinarse éstos con otros mecanismos para diseñar algoritmos de localización de recursos en redes cuyo rendimiento supera al de las búsquedas basadas en caminos aleatorios normales en cuanto a la longitud media de la búsqueda.

En particular, se ha demostrado que esto es cierto para los mecanismos presentados, basados en caminos aleatorios SAW y en caminos aleatorios parciales, y aplicados a redes construidas aleatoriamente y en las que se cumple que cualquier punto de conexión de un nodo puede estar conectado mediante un enlace a cualquier otro punto de conexión de la red con probabilidad uniforme.

Este trabajo puede extenderse a lo largo de diversas líneas, con el objetivo de obtener mecanismos de búsqueda de recursos más eficientes:

- Modificar las reglas de elección del siguiente nodo en el camino aleatorio, tomando en consideración, por ejemplo, el grado de los vecinos, el número de veces que el vecino ha sido visitado o cubierto, el número de recursos que posee, etc.

- Modificar los criterios de selección de un camino parcial en el nodo actual. Por ejemplo, si ninguno de ellos posee el recurso deseado, podría elegirse el camino parcial con mayor grado medio de sus nodos.

- Analizar y evaluar el rendimiento de los mecanismos basados en caminos aleatorios parciales cuando el uso de estructuras de datos probabilísticas y los recursos dinámicos se combinan como fuentes de falsos positivos.

- Analizar y evaluar el rendimiento de los mecanismos de búsqueda en redes con compartamiento dinámico de sus nodos (entradas y salidas de nodos en la red). Un enfoque complementario sería estudiar el efecto de enlaces dinámicos (los nodos permanecen pero los enlaces van cambiando con el tiempo).

- Finalmente, sería interesante hacer una comparación detallada de los beneficios y costes de los mecanismos presentados y de las soluciones existentes para systemas estructurados (por ejemplo, DHT).

Appendix C

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image

format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute.

However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling

the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on.

These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b)

permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.