Hybrid Search Schemes for Unstructured Peer-to-Peer Networks

IEEE Infocom 2005

Christos Gkantsidis, Milena Mihail
College of Computing
Georgia Institute of Technology
Atlanta, GA, 30332, USA
Email: {gantsich, mihail}@cc.gatech.edu

Amin Saberi
Institute for Computational and Mathematical Engineering
and Management Science and Engineering Department
Stanford University
Stanford, CA, 94305, USA
Email: saberi@stanford.edu

***Abstract*— We study hybrid search schemes for unstructured peer-to-peer networks. We quantify performance in terms of number of hits, network overhead, and response time. Our schemes combine flooding and random walks, look ahead and replication. We consider both regular topologies and topologies with supernodes. We introduce a general search scheme, of which flooding and random walks are special instances, and show how to use locally maintained network information to improve the performance of searching. Our main findings are: (a)A small number of supernodes in an otherwise regular topology can offer sharp savings in the performance of search, both in the case of search by flooding and search by random walk, particularly when it is combined with 1-step replication. We quantify, analytically and experimentally, that the reason of these savings is that the search is biased towards nodes that yield more information. (b)There is a generalization of search, of which flooding and random walk are special instances, which may take further advantage of locally maintained network information, and yield better performance than both flooding and random walk in clustered topologies. The method determines edge criticality and is reminiscent of fundamental heuristics from the area of approximation algorithms.**

I. Introduction

Flooding is the predominant search technique in unstructured peer-to-peer (P2P) networks. If we measure performance as the number of exchanged messages per distinct response, flooding with small time-to-live performs well in regular networks. However, its performance deteriorates as the time-to-live increases, or if the topology of the underlying network is not regular [1]. In addition, flooding has poor granularity [2], [3].

Simulating a random walk has been proposed as an alternative search technique. In regular topologies, the performance of the random walk simulation method appears to be better than the performance of flooding. In addition, the random walk simulation method scales well and has excellent granularity. However, the simulation of a random walk is inherently sequential, which causes a large increase in the response time [1], [3].

We consider hybrid schemes which can be viewed as a random walk of substantially shorter length (and hence smaller response time), combined with very shallow floodings on every step of the random walk (see Figure 1d); similar

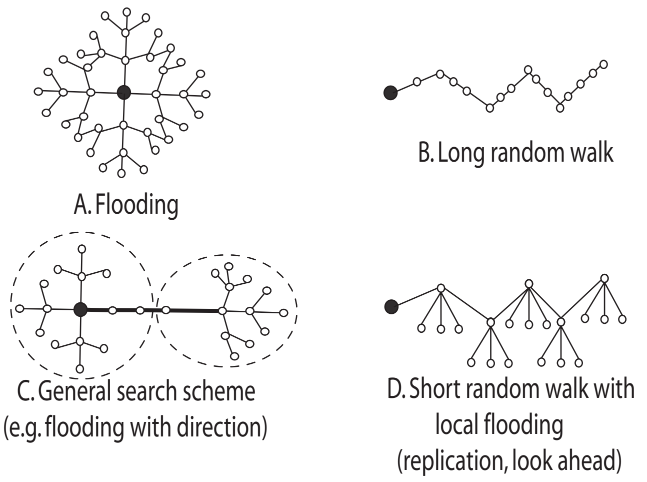
A. Flooding
B. Long random walk
C. General search scheme
(e.g. flooding with direction)
D. Short random walk with
local flooding
(replication, look ahead)

Fig. 1. Figure 1a represents search by flooding. Flooding has good performance for small values of time-to-live. Figure 1b represents search by random walk. The response time is proportional to the length of the walk. Figure 1c represents a general search scheme, which is flooding amplified towards a critical direction. This is suitable in the case of clustered topologies, where the critical direction leads flooding outside a cluster. Figure 1d represents a shorter random walk with local floodings. This decreases the response time and is particularly suitable when combined with 1-step replication.

hybrid schemes have been discussed explicitly in [4], [5] and implicitly in [6]. We shall refer to these schemes as random walks with lookahead. Alternatively, very shallow floodings, say of depth 1, can be thought of as a 1-step replication strategy, that is, where each node keeps a copy of the indices of his neighbors. In sparse networks, such replication causes low network overhead, while the benefits of the replication can be enjoyed by all future searches.

What is the analytic justification for the good performance of such hybrid schemes? Does the analysis suggest further efficient search algorithms? Is there a general abstraction, of which flooding and random walks are special instances? Can such an abstraction be useful in obtaining even more efficient search algorithms?

The first contribution of this paper is to give analytic justification of why the simulation of a short random walk with shallow floodings on every step performs particularly well. The idea is the following. Naturally, we expect that the time to discover a certain number of nodes using a random walk with shallow floodings will be somewhat smaller than in the simulation of a random walk without local floodings; the reason is that in each step of the random walk with shallow floodings we visit a node and all its neighbors, In particular, in a sparse network (say, with constant average degree and hence constant average gain per node), we would intuitively expect a constant saving in the response time. We show that there are sparse networks where the saving in the response time can be much sharper. In particular, we show that, in a standard random graph model, if the network has $\Theta(n)$ nodes of constant degree and $\Theta(\sqrt{n})$ nodes of degree $\Theta(\sqrt{n})$ then, the expected time to discover $\Omega(n)$ nodes is $O(\sqrt{n})$; this is in Theorem 4.2. The proof indicates that the reason for the dramatic improvement in the response time of random walk with local floodings is because the random walk biases the sampling towards the nodes that have high degree and hence yield more information using the shallow floodings. *From the practical point of view, Theorem 4.2 suggests that search by random walk with lookahead 1, or with 1-step replication, substantially amplifies its benefits when there is discrepancy on the degrees of the underlying topology.* (A remark is due about the assumption of discrepancy of $\Theta(\sqrt{n})$ in the degrees of a P2P network. Is this assumption realistic? Firstly, note that we show our results for large degrees $\beta\sqrt{n}$, where β can be a small constant; one may interpret the degrees of ultra-peers in current P2P networks for some $\beta = 0.1$. Secondly, we mainly use the $\Theta(\sqrt{n})$ assumption in our analytical results, for which this choice makes the calculations and the principal underlying phenomena cleaner. We can obtain similar results for much smaller values of the large degrees, with much more detailed calculations.)

On the other hand, we noted that flooding has poor performance when there is discrepancy in the degrees of the underlying network. The second contribution of this paper is to rectify the performance of flooding in the case of a sparse network with a few vertices of large degrees. We study normalized flooding, where a vertex of small degree forwards a query to all his neighbors, while a vertex of large degree forwards a query to a small subset of its neighbors chosen uniformly at random. In Theorem 3.3, we show that, in a random network with $\Theta(n)$ nodes of constant degree and $\Theta(\sqrt{n})$ nodes of degree $\Theta(\sqrt{n})$, normalized flooding achieves performance comparable to flooding in a regular graph. In Theorem 4.3, we further show that normalized flooding with 1-step replication achieves performance comparable to random walk with 1-step replication, further indicating that the gaining in 1-step replication comes from the bias of large degrees, and further strengthening the suggestion to use a small number of supernodes.

The moral of Theorems 4.2 and 4.3 can be thought of as follows. These theorems tell us that, *using local information of the network, in this case the degrees, we can get global benefit, by biasing the sampling towards the vertices with a lot of information*. Is there other local information that can be useful in searching? In a long sequence of theory papers, information concerning "edge criticality" has given novel approximation algorithms for NP-complete problems. In particular, [7] show that there are labels that can be assigned to edges, so that edges across bad cuts of the graph get heavier weights, and doing region growing according to these labels finds sparse cuts [8]. In addition, very roughly, [9], [10] show that these labels can be approximated by (repeated computations of) congestion under shortest path routings. Reminiscent of these techniques, *we define edge criticality metrics that identify edges belonging to sparse cuts*. We note that these metrics can be computed by local statistics that the network keeps anyway.

The third contribution of this paper is to *define a generalization of searching,* of which flooding, random walks and random walks with lookahead are special instances. This is particularly simple to implement. In particular, we assume that a node initiates a search by assigning a budget, which is an upper bound on the number of messages that will be exchanged during the search. The node then partitions the budget, and may forward different partition classes of the budget to different neighbors. We show that this scheme is *particularly useful when edge criticality is known*. See Figure 1c. Suppose that the underlying network is clustered. Then, the thick edges will be assigned larger criticality, thus shifting a substantial part of the initial budget outside a specific cluster, and essentially initiate a new flooding in a different cluster. We report experimentation, where this heuristic has very good performance.

In summary, we show that (a) the existence of super-nodes improves searching performance if combined with suitable defined protocols, and (b) preferential treatment of links according to their criticality can further improve protocol performance.

The balance of the paper is as follows. In Section II we give the graph models that we use and some crucial structural properties that will be later used in the proofs. In Section III we review the good behavior of flooding in regular graphs (Theorem 3.1), and argue that this behavior deteriorates in graphs with supernodes (Theorem 3.2) while normalization rectifies this deterioration (Theorem 3.3). In Section IV we review the behavior of random walks in regular graphs (Theorem 4.1), and argue that 1-step replication in graphs with supernodes can substantially improve the performance of the random walk method (Theorem 4.2). We further show that similar savings can be achieved by normalized flooding and 1-step replication (Theorem 4.3), further indicating that the savings come from the use of supernodes. In Section V we describe the generalized search scheme. In Section VI we report experimental evaluation.

II. Random Graph Models

In this section we introduce random regular graphs, which aim to capture the behavior of a typical regular topology, and random graphs with supernodes, which aim to capture the typical behavior of a sparse network with a small number of large nodes. We review the graph theoretic notion of expansion and relate to the performance of flooding (see expressions 3, 5, and 1).

For graphs with supernodes, we establish new structural facts which characterize the neighborhoods of nodes with large degrees (Lemmas 2.1, 2.2 and 2.3).

We use the configurational random graph model. This is a standard model in the theory of random graphs as well as networking. In particular, for $d_1 \geq d_2 \geq \ldots \geq d_n$ denoting the degrees of a graph on n nodes, we generate a random graph as follows. First consider $D = \sum_{i=1}^{n} d_i$ mini-vertices corresponding to nodes in the natural way: the first d_1 mini-vertices correspond to node 1, the next d_2 mini-vertices correspond to node 2, and so on. Then consider a random perfect matching over the D mini-vertices, and a graph on the original n vertices defined by adding one link from node i to node j for each edge of the perfect matching that was connecting a mini-vertex corresponding to i with a mini-vertex corresponding to j. Note that this is a multigraph with self loops. In this section we maintain multiple links and self loops for analytic convenience.

Let d be a constant. By *random regular graph*, denoted $G_{n,d}$, we mean a random graph in the configurational model, with $d_i = d$, $1 \leq i \leq n$. We next introduce a random graph model for graphs with supernodes. Let α and β be constants. Consider $\alpha n^{\frac{1}{2}}$ nodes of degree $\beta n^{\frac{1}{2}}$, called *large vertices*, and all the remaining nodes of degree d, called *small vertices*. By *random graph with supernodes*, denoted $G_{n,d,\alpha,\beta}$, we mean a random graph in the configurational model following the above degree sequence. Note that random regular graphs and random graphs with supernodes have sum of degrees $D = dn$ and $D \simeq (\alpha\beta + d)n$ respectively, hence they are sparse, in the sense that the sum of the degrees of their vertices is $\Theta(n)$. Throughout this paper, $\simeq$ means $1 \pm o(1)$.

We further review the notion of vertex neighborhood and relate it to the performance of flooding. All the theorems in Section III are based on the characterization of vertex neighborhoods. We need the following definitions. Let $G(V,E)$ be an undirected graph, with $|V| = n$. Let S be a subset of vertices, $S \subset V$, and let $\bar{S}$ be its complement, $\bar{S} = V \setminus S$. Define the *vertex neighborhood* of S as $\Gamma(S) = \{u \in \bar{S} : (v,u) \in E\text{, for some } v \in S\}$. Now let $S_i(v)$ be the set of vertices visited by flooding that initiated at vertex v with time-to-live i, and note that $S_i(v) = S_{i-1}(v) \cup \Gamma(S_{i-1}(v))$. Suppose that G is a d-regular graph. How many messages did each vertex propagate? The vertex v propagated d messages, and each vertex in $S_{i-1}(v)$ propagated at most $d-1$ messages, namely to all its neighbors except the one from which he received the query. Vertices in $\Gamma(S_{i-1}(v))$ were reached with time-to-live 0 and hence did not propagate any messages. Now we may upper bound the ratio of distinct responses over number of messages

$$\frac{|S_{i-1}| + |\Gamma(S_{i-1}(v))|}{(d-1)|S_{i-1}(v)|} = \frac{1}{d-1}\left(1 + \frac{|\Gamma(S_{i-1}(v))|}{|S_{i-1}(v)|}\right) \quad (1)$$

Since, 1 describes the performance of searching, it is important to examine the ratio $\Gamma(S_{i-1}(v))/|S_{i-1}(v)|$.

For a random regular graph $G_{n,d}$, [11] show the following property. For an arbitrary subset of vertices S of a graph G, define the *cutset* of S, $\nabla(S)$, as $\nabla(S) = \{(v,u) \in E : v \in S, u \in \bar{S}\}$. Let $S_i(v)$ be the set of vertices reached by flooding with time-to-live i, as above. [11] show that, with probability $1 - o(n^{-2})$, for all vertices v and for all $i : (d-1)^i \leq n^{\frac{1}{2}} \log n$,

$$|\nabla(S_i(v))| \geq\simeq (d-1)^i \quad (2)$$

We claim that this implies that, for all vertices v, and for all $i : (d-1)^i \leq n^{\frac{1}{2}}$,

$$|\Gamma(S_i(v))| \geq\simeq (d-1)^i \quad (3)$$

To see this, note that (2) applied to $S_{i+1}(v)$ gives $|\nabla(S_{i+1}(v))| \geq\simeq (d-1)^{i+1}$. But all edges in $\nabla(S_{i+1}(v))$ must be incident to a vertex in $\Gamma(S_i(v))$, and each vertex in $\Gamma(S_i(v))$ can yield at most $d-1$ edges in $\nabla(S_{i+1}(v))$, which implies that $|\Gamma(S_i(v))| \geq\simeq (d-1)^i$.

For the random graph $G_{n,d}$, [12] further claim:

$$|\nabla(S)| \geq \left\{ \begin{array}{ll} (1 - O(\frac{1}{\sqrt{d}} + \epsilon))d|S| & |S| \leq \epsilon|V| \\ d|S|/4 & \epsilon|V| \leq |S| \leq |V|/2 \end{array} \right. \quad (4)$$

which immediately implies

$$|\Gamma(S)| \geq \left\{ \begin{array}{ll} (1 - O(\frac{1}{\sqrt{d}} + \epsilon))|S| & |S| \leq \epsilon|V| \\ |S|/4 & \epsilon|V| \leq |S| \leq |V|/2 \end{array} \right. \quad (5)$$

We will use (3) and (5) in the characterization of flooding in Section III.

We proceed to discuss structural properties for graphs with supernodes. The crucial structural properties are that each small node is incident to a large degree node with constant probability, and each large node has, in expectation and with sharp concentration, a constant fraction of its edges incident to distinct large nodes and a constant fraction of its edges incident to small degree nodes. We will also use the following form of Chernoff bounds [13]. Let X_i, $i = 1, \ldots, N$, be independent random variables with $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$. Let $X = \sum_{i=1}^{N} X_i$ and let $p = (\sum_{i=1} N p_i)/n$. Then,

$$\Pr[X - pN < -\Delta] < e^{-\frac{\Delta^2}{2pN}} \quad (6)$$

and

$$\Pr[X - pN > \Delta] < e^{\frac{\Delta^2}{2pN} + \frac{\Delta^3}{2(pN)^3}} \quad . \quad (7)$$

More formally, the structural facts for graphs with supernodes are Lemmas 2.1, 2.2 and 2.3 below.

Lemma 2.1: Let $G = G_{n,d,\alpha,\beta}$ be a random graph with supernodes, and let ϵ be any constant $\epsilon < \max\{\alpha, \beta\}$. Then, with all but exponentially vanishing probability, every large vertex of G has $\frac{(\alpha-\epsilon)\beta}{d+\alpha\beta}\frac{\epsilon n^{\frac{1}{2}}}{2}$ distinct large neighbors.

Proof: Let v be a large vertex. Suppose that $N = \epsilon n^{\frac{1}{2}}$ distinct neighbors of v are known to be distinct large vertices. What is the probability that v is incident to an additional distinct large vertex? There are $(\alpha - \epsilon)n^{\frac{1}{2}}$ remaining large vertices, hence the remaining total degree on large vertices is $(\alpha-\epsilon)n^{\frac{1}{2}}\beta n^{\frac{1}{2}}$. The total degree of all vertices is $(d+\alpha\beta)n$. Hence the probability that v sees an additional distinct large degree vertex, given that it has seen $\epsilon n^{\frac{1}{2}}$ distinct vertices is at least $p = \frac{(\alpha-\epsilon)\beta}{d+\alpha\beta}$. We may now bound the probability that v is incident to less than $pN/2$ distinct vertices by observing that this probability is smaller than the probability in N independent experiments, each with probability of success p, there were less than $\Delta = pN/2$ successes. Using (6) above, we get exponentially small tails. ■

Lemma 2.2: Let $G = G_{n,d,\alpha,\beta}$ be a random graph with supernodes. Then, with all but exponentially vanishing probability, every large vertex of G has $\frac{d}{d+\alpha\beta}\frac{\beta n^{\frac{1}{2}}}{2}$ edges incident to (not necessarily distinct) small neighbors.

Proof: Let v be a large vertex. Let $N = \beta n^{\frac{1}{2}}$. Suppose that $N-1$ edges incident to v are known to have their other endpoint incident to a small vertex. Then, the probability that the last edge is also incident to a small vertex is $\frac{dn-d}{(d+\alpha\beta)n} \simeq \frac{d}{d+\alpha\beta} = p$. We may now bound the probability that v has less than $\frac{d}{d+\alpha\beta}\frac{\beta n^{\frac{1}{2}}}{2}$ edges incident to small vertices by the probability that in N independent experiments, each with probability of success p, there were fewer than $\Delta = pN/2$ successes. By (6), this is exponentially small. ■

Lemma 2.3: Let $G = G_{n,d,\alpha,\beta}$ be a random graph with supernodes. Then, with all but exponentially vanishing probability, every large vertex v has $\Gamma(\{v\}) \cup \Gamma\left(\Gamma(\{v\})\right) \geq \frac{(\alpha-\epsilon)\epsilon\beta^2}{4(d+\alpha\beta)^2}n$.

Proof: By Lemma 2.1, v has $\frac{\alpha-\epsilon}{d+\alpha\beta}\frac{\epsilon\beta n^{\frac{1}{2}}}{2}$ distinct large neighbors. By Lemma 2.2, each distinct large neighbor has $\frac{d}{d+\alpha\beta}\frac{\beta n^{\frac{1}{2}}}{2}$ distinct edges incident to small vertices. But each small vertex has at most d incident edges, and the statement of the claim follows. ■

III. Flooding and Normalization

Flooding is the predominant search technique in unstructured peer-to-peer networks. Such floodings are typically parameterized by a time-to-live, τ. In particular, a node initiates a search by propagating a request, together with a time-to-live τ, to all his neighbors. Without loss of generality, we may think of the request as an exploration of the network: "if you get this message for the first time, then report your presence (e.g. address) to the initiator of the request". Flooding proceeds as follows. The first time that a node receives a request with time-to-live t, the node responds to the request and, if $t > 0$, the node propagates the same request to all his neighbors. If a node receives the same request multiple times, then it will neither respond nor propagate it.

We quantify the performance of flooding by the *number of responses*, the *response time* (we assume that the delay of a particular response is proportional to the number of hops between the initiator of the query and the responding node), and by the *number of propagated messages*. Clearly, the number of responses and the response time quantify the quality of service perceived by the initiator of the search, and the number of propagated messages quantify the overhead perceived by the network. In practice, flooding is known to perform very well for small values of τ, however the performance does not scale well with τ. In addition flooding has poor *granularity*.

When a graph is not regular, then the performance of flooding deteriorates. In particular, when large degree vertices are reached, then these cause a sudden sharp increase in the number of neighbors they introduce (hence poor granularity), which, in turn, causes a lot of shared edges (hence poor performance in terms of number of messages per distinct number of discovered nodes). We therefore consider *normalized flooding* which is the following algorithm. Let $d_{\min}$ be the minimum degree of the network. In normalized flooding, when a node of degree $d_{\min}$ receives a query, the node propagates the query to all his neighbors (except the one which forwarded the query). When however a node of larger degree receives a query, the node propagates the query only to $d_{\min}$ of his neighbors, chosen uniformly at random from the entire set of his neighbors (except the one which forwarded the query). This is the natural normalization, and it is well known common practice (e.g. see [14]).

In Theorem 3.1 we establish the good behavior of flooding in regular graphs. The proof of this theorem is directly based on known structural properties of random regular graphs. Our contribution is to translate these properties in the context of flooding, as expressed in (1). It is important to notice that the upper bounds in Theorem 3.1 differentiate between ranges of the time-to-live, and clearly suggest that the guarantees on the performance of flooding deteriorate as the time-to-live increases. The number of distinct nodes discovered increases exponentially, and this indicates poor granularity.

Theorem 3.2 is a lower bound for flooding in graphs with supernodes. It indicates that, without normalization, a large vertex is discovered for a very small value of time-to-live, hence even poorer granularity. Theorem 3.3 indicates that normalized flooding in graphs with supernodes can rectify the performance of flooding. In particular, it brings the performance of normalized flooding, up to order of magnitude, to the performance of flooding in regular graphs (we show this for small values of time-to-live where flooding in regular graphs has its best behavior). The proofs of Theorems 3.2 and 3.3 make critical use of the structural properties established in Lemmas 2.1, 2.2, and 2.3.

For analytic convenience in the analysis of normalized flooding, we think of the following finer structure in $G_{n,d,\alpha,\beta}$. Each set of $\beta n^{\frac{1}{2}}$ minivertices corresponding to a large vertex is further partitioned into minigroups of d minivertices. We may now think of $G_{n,d,\alpha,\beta}$ as a random regular graph with all minigroups corresponding to the same large vertex contracted to a single large vertex.

Theorem 3.1: Let $G_{n,d}$ be a random regular graph, let v be a node of $G_{n,d}$, and consider a flooding in the basic scenario initiated by v with time-to-live τ. Let S be the number of distinct nodes queried by this flooding and suppose that $|S| \leq |V|/2$. Then, for $\tau \leq \frac{\log n}{2\log(d-1)}$, the number of distinct responses is $|S| \geq\simeq (d-1)^{\tau-1}$ and the number of distinct responses per message is at least $\simeq \frac{1}{d-1}\left(2-O(\frac{1}{\sqrt{d}})\right)$, almost surely. Furthermore, for any S with $|S| \leq \epsilon|V|$, $\epsilon < 1/2$, the number of distinct responses is $|S| \geq\simeq \left(2-O(\frac{1}{\sqrt{d}}+\epsilon)\right)^{\tau}$ and the number of distinct responses per message is at least $\simeq \frac{1}{d-1}\left(2-O(\frac{1}{\sqrt{d}})\right)$, almost surely. Finally, for any S with $|S| \leq |V|/2$, the number of distinct responses is $|S| \geq\simeq (1+\frac{1}{4})^{\tau}$ and the number of distinct responses per message is at least $\frac{1}{d-1}(1+\gamma)$, almost surely.

Proof: For random regular graphs, the behavior of flooding for $\tau \leq \frac{\log n}{2\log(d-1)}$ is derived from the fact that breadth-first-search with bounded depth, in particular until $n^{\frac{1}{2}}\log n$ nodes are visited, has very good behavior, almost surely. We expressed this in formula (3). Now the performance claimed in Fact 3.1 for $\tau \leq \frac{\log n}{2(d-1)}$ can be obtained as follows. Using (3), the number of distinct responses received with time-to-live τ is

$$\begin{array}{rcl} \sum_{i=0}^{\tau-1}|\Gamma(S_i(v))| & \geq & \simeq \sum_{i=0}^{\tau-1}(d-1)^i \\ & = & \frac{(d-1)^{\tau+1}-1}{d-2} \\ & \geq & (d-1)^{\tau} \ . \end{array} \quad (8)$$

For the number of messages per distinct response we use (5). Now the number of messages per distinct response follows by substituting (5) in (1). In particular, since $|S| \leq (d-1)^{\tau}$, for $\tau \leq \frac{\log n}{2\log(d-1)}$ we get $|S| \leq |V|^{\frac{1}{2}}$, which implies $\epsilon \leq |V|^{-\frac{1}{2}}$, and hence $|\Gamma(S_{\tau-1}(v))|/|S_{\tau-1}(v)|$ is at least $1-O(1/\sqrt{d})$. ■

Theorem 3.2: Let $G_{n,d,\alpha,\beta}$ be a random graph with supernodes, let v be a node of $G_{n,d,\alpha,\beta}$ of degree d, and consider a flooding initiated by v in the basic flooding scenario. Then, for some time-to-live $\tau = \Theta(\log\log n)$, the number of distinct responses is $\Omega(n)$, almost surely.

Proof: Consider flooding with time-to-live $\tau \simeq c\log_{d-1}\log n + 1$, for some constant c. We first argue that, with all but polynomially vanishing probability, a large vertex is found. To see this, consider the vertices visited with time-to-live up to τ–1, and suppose that this set does not contain a large degree vertex. This set then can be thought of as the result of flooding on a random d-regular graph, and by (2), the cutset of this set has at least $(d-1)^{\tau-1}$ edges. The probability that the other endpoint of each edge in this cutset is a small vertex is $\frac{d}{d+\alpha\beta}$. Thus the probability that no vertex in $\Gamma(S_{\tau-1}(v))$ is large can be bounded by $(\frac{d}{d+\alpha\beta})^{(d-1)^{\tau-1}} = (\frac{d}{d+\alpha\beta})^{c\log n}$. So we know that, almost surely, within the first $O(\log\log n)$ steps we will see a large vertex. Now, by Lemma 2.3 this vertex will explore $\Omega(n)$ vertices in two more steps of the flooding. ■

Theorem 3.3: Let $G_{n,d,\alpha,\beta}$ be a random graph with supernodes, let v be a node of $G_{n,d,\alpha,\beta}$, and consider a normalized flooding initiated by v with time-to-live $\tau \leq \frac{\log n}{2\log(d-1)}$. Then, the number of distinct responses is $\Omega((d-1)^{\tau-1})$ and the number of messages per response is $O(1)$, almost surely.

Proof: By Theorem 3.1, in τ, the number of minigroups seen is $(d-1)^{\tau-1}$. The expected number of small vertices is $\frac{d}{(d+\alpha\beta)}(d-1)^{\tau-1}$. Now using (6), the probability that less than $\frac{d}{2(d+\alpha\beta)}(d-1)^{\tau-1}$ are seen is vanishingly small. ■

IV. Random Walks and Replication

Everything else being equal, the best way to search a graph would be by uniform sampling. Assuming that a random node of the network could be generated efficiently, we could take k such samples simultaneously at cost one message per sample. By the well known coupon collection theorem (uniform sampling with replacement), for any $1 \leq k \leq n$, the expected number of samples to visit k distinct nodes is $(H_n - H_{n-k})n$, where H_i is the i-th harmonic number. In particular, the expected number of samples to visit all the nodes is $n\log n$ and, for any constant $\epsilon < 1$, the expected number of samples to visit ϵn distinct nodes is $\frac{\epsilon}{1-\epsilon}n$. Thus, the amount of network overhead per distinct response can come arbitrarily close to 1, while retrieving a constant fraction of the search space. In addition, all the samples can be drawn simultaneously. The drawback of course is that it is not known how to implement uniform sampling in the relevant application context.

The *random walk* method has been proposed as a practical alternative to implement uniform sampling [1], [3]. In particular, in several random graph models, the so-called mixing time of the random walk, which is the number of simulation steps in order for the random walk to reach a distribution close (for sampling purposes) to uniform, is $O(\log n)$. This means that we may simulate k uniform samples with $O(\log n)$ random walk steps for each uniform sample. Since the random walks can be simulated in parallel, and assuming that the response delay of a random walk is proportional to the number of simulation steps of the walk, we get maximum response time $O(\log n)$, overhead at most $O(k\log n)$, while achieving performance similar to uniform sampling. The drawback of this approach is the network overhead which scales as $O(\log n)$. On the positive side, the theory of cover times [15] [16], complexity theory [17], [18] and extensive experimentation [1] suggest that this overhead can be reduced to a constant by taking $O(\log n)$ steps to randomize and then using k *successive steps* of the random walk in the place of independent samples. The drawback however is that the approach is inherently sequential and hence introduces maximum response time at least k.

The behavior of the random walk method for regular graphs is in Theorem 4.1 below. We give this well known theorem without proofs.

Theorem 4.1: Let $G_{n,d}$ be a random regular graph, let v be a node of $G_{n,d}$, and consider a random walk starting at v. Then, for any k with $1 \leq k < n$, the expected number of messages and response time to get k distinct responses is at most $(H_n - H_{n-k})nO(\log n)$, almost surely. In addition, the expected number of messages to get n distinct responses is $\frac{d-1}{d-2} n \log n$, almost surely.

One way to reduce the response time is to perform a much shorter walk, and in addition perform shallow floodings on each step of the walk. We call this method *random walk with lookahead*. In regular graphs, for constant lookahead (flooding with constant depth) we expect a constant saving in the response time. Here we observe that the savings in the response time are much sharper, if the graph has supernodes; similar results have been also observed in [4] and [5] in power law graphs. In particular, Theorem 4.2 suggests that, for lookahead 1, we may visit $\Omega(n)$ nodes with response time $O(n^{\frac{1}{2}})$. The proof of Theorem 4.2 makes crucial use of the structural properties of graphs with supernodes that were established in Section II.

Let us further consider 1-step replication. In this scenario, a node maintains information about all his neighbors and, when queried, includes this information in his response. In experiment, [3] observed very good performance of the sequential random walk method in a network with 1-step replication. In a sparse network, 1-step replication can be implemented with a one-time linear overhead where all edges exchange the information of their endpoints, while the benefit of this replication can be experienced by all future queries (see also [19] for another application of lookahead). Theorem 4.2 establishes the performance of 1-step replication. Realize that lookahead and 1-step replication are different implementations of the notion of short random walks with flooding with time-to-live 1 at every step.

Intuitively, the reason why lookahead and 1-step replication offer very sharp savings in graphs with supernodes is that the stationary distribution of the random walk has sharp bias towards large vertices, which yield a lot of information about their neighbors. Is random walk the only way to achieve such savings? In Theorem 4.3 we show that normalized flooding can achieve similar savings (up to order of magnitude). Intuitively, the reason is that normalized flooding can be also thought of as mimicking sampling from a distribution with sharp bias towards large vertices.

Theorem 4.2: Consider any ϵ such that $0 \leq \epsilon < \frac{1}{2}$. Let $G_{n,d,\alpha,\beta}$ be a random graph with supernodes, let v be a node of $G_{n,d,\alpha,\beta}$, and consider a random walk starting at v. Then, in the 1-step replication scenario, the expected number of messages and response time to obtain $\frac{\alpha\beta n^{1-\epsilon}}{4d} = \Omega(n^{1-\epsilon})$ distinct responses is $\frac{d+\alpha\beta}{\alpha\beta} O(\log n) \frac{\alpha n^{\frac{1}{2}-\epsilon}}{2} = O(n^{\frac{1}{2}-\epsilon} \log n)$, almost surely.

Proof: The stationary distribution of the random walk is as follows. Each large vertex has probability $\beta/(d+\alpha\beta)\sqrt{n}$, and each small vertex has probability $d/(d+\alpha\beta)n$. Since there are $\alpha\sqrt{n}$ large vertices, the stationary probability of all large vertices is $\frac{\alpha\beta}{d+\alpha\beta}$. Now for the simulation of the random walk, using the conductance result of [20], we get that, in $O(\log n)$ simulation steps we will have a vertex sampled from a distribution which is arbitrarily close to the stationary. Hence, in expected $\frac{d+\alpha\beta}{\alpha\beta} O(\log n)$ simulation steps we get a large vertex, and, by coupon collection, in expected $\sum_{j=1}^{\alpha n^{\frac{1}{2}-\epsilon}/2} \frac{\alpha n^{\frac{1}{2}}}{\alpha n^{\frac{1}{2}} - j + 1} = \frac{\alpha n^{\frac{1}{2}-\epsilon}}{2}$ large vertices we get $\frac{\alpha n^{\frac{1}{2}-\epsilon}}{2}$ distinct large vertices. So in expected $\frac{d+\alpha\beta}{\alpha\beta} O(\log n) \frac{\alpha n^{\frac{1}{2}-\epsilon}}{2}$ simulation steps we get $\frac{\alpha n^{\frac{1}{2}-\epsilon}}{2}$ distinct large vertices. Since each large vertex has $\frac{\beta n^{\frac{1}{2}}}{2}$ edges incident to small vertices, we get $\frac{\alpha\beta n^{1-\epsilon}}{4}$ edges incident to small vertices. But each small vertex can be incident to at most d large vertices, which completes the proof ■

Theorem 4.3: Let $G_{n,d,\alpha,\beta}$ be a random graph with supernodes, let v be a node of $G_{n,d,\alpha,\beta}$. Consider normalized flooding starting at v with time-to-live $\tau \simeq \frac{\log n}{2\log(d-1)}$. Then, in the 1-step replication scenario, the number of distinct responses is at least $\frac{(d-1)^{\tau-1}\alpha b^2 n^{\frac{1}{2}}}{8d(d+\alpha\beta)} = \Omega(n)$, almost surely, and the number of messages is at most $(2 - O(\frac{1}{\sqrt{d}}))(d-1)^{\tau} = O(n^{\frac{1}{2}})$.

Proof: By reasoning as in the proof of Theorem 3.1, there will be $(d-1)^{\tau-1}$ minigroups. Using (6), there will be $\frac{(d-1)^{\tau-1}\alpha\beta}{2(d+\alpha\beta)}$ minigroups corresponding to large vertices. How many minigroups corresponding to distinct large vertices were found? The probability that a group found corresponded to a distinct large vertex is at least $\frac{\alpha n^{\frac{1}{2}} - \frac{(d-1)^{\tau-1}\alpha\beta}{2(d+\alpha\beta)}}{\alpha n^{\frac{1}{2}}} \geq 1/2$. Using (6), there will be $\frac{(d-1)^{\tau-1}\alpha\beta}{4(d+\alpha\beta)}$ distinct large vertices. Now, using Lemma 2.2, each distinct large vertex has $\frac{\beta n^{\frac{1}{2}}}{2}$ incident small vertices, and since each small vertex can be incident to at most d large vertices, we get a total of at least $\frac{(d-1)^{\tau-1}\alpha b^2 n^{\frac{1}{2}}}{8d(d+\alpha\beta)}$ distinct small vertices. ■

V. Generalized Search Schemes

We now describe a new searching scheme that allows very fine granularity of the number of messages that will be used for searching, like searching with random walks, and still allows very fast searching, like searching with flooding. A node initiates a search by picking a *budget* k, which is the number of messages that will propagate in the network. Assuming that the node has d neighbors, then the node distributes its budget by picking d integers $k_1, \ldots, k_d$, with $k_i \geq 0$, $1 \leq i \leq d$, and $k_1 + \ldots + k_d = k$. Then, it forwards the query to node i with budget equal to k_i (if $k_i = 0$ then the query is not forwarded to node i). Each neighbor i will reduce the budget received by 1 and repeat the same process if the new budget is greater than 0. Because the generalized searching is sensitive to budgets, if a node receives the same query for a second time, from a different neighbor, then it will forwarded it again. Of course, the most critical task is the choice of k_1 to k_d.

The generalized searching scheme has both random walks and floodings as a special instance. To simulate random walk each node picks a neighbor i at random and assigns to it the remaining budget (say $k-1$); all other neighbors are assigned a budget of 0 and thus no query message is forwarded to them. Flooding in regular graphs can also be simulated easily. Each node divides the budget equally to all its neighbors minus the neighbor from which it received the query (if it did not initiate the message itself). To compute the initial budget, the node initiating the query needs to have an estimate of the number of messages that a regular flooding with TTL τ would generate; in regular graphs with degree d this can be $(d-1)^{\tau}$. In general graphs it is not possible to simulate flooding exactly. A good approximation however is to divide the budget to each neighbor according to their degrees.

The main advantage of the generalized searching is that it allows arbitrary assignment of budget to each neighbor. Intuitively, a node should assign a larger budget to neighbors through which more peers can be reached. This is particularly important for topologies that have clusters. Assume a topology with two clusters and few edges between the clusters. When the search reaches a border node, then that node needs to forward the query with larger budget to the other cluster since the nodes in its own cluster can be reached from different paths. Moreover, the query should propagate with higher weight towards the border nodes. How should a node allocate the budget to its neighbors to achieve that behavior? In other words, what are good heuristics to compute the k_i's?

Let us use the generic term "edge criticality" to denote metrics related to the importance of the edges. One way to define the criticality of an edge is as the number of shortest-hop paths between any pair of nodes of the network that use that edge. The few inter-cluster edges and the edges that lead to them are assigned higher criticality since a large number of paths will go through them. Thus, an effective way to perform searching in topologies with clusters is to divide the budget according to the criticality of the edge that connects to each neighbor.

Computing the edge criticality, according to the shortest-hop path definition, is extremely difficult since each node should perform a flooding to the entire network to discover the shortest-hop paths to each other node. A more practical approach is to have only a subset of nodes discover the shortest hop paths to each node. Experimentally, we have observed that computing the shortest-paths from less than 2% of the nodes was sufficient. Another practical approach is to discover the shortest-hop paths to all other nodes that have a distance less than a specified value. In current P2P network, like Gnutella, each node periodically floods the network with TTL 7 to discover the peers that are in its horizon. If the flooding message contained the identity of the originating node, then intermediate nodes could monitor the queries and the replies to infer how many shortest-paths go through them. In general, estimating the edge criticality is possible in current networks, and, moreover, for the purpose of searching with budgets, a rough estimation of the edge criticality gives very good results as we shall see later in Section VI-E.

Note that current peer-to-peer networks already implement heuristics that use information collected locally to improve the performance of the network. In implementations of the gnutella protocol, including mutella [21], the nodes estimate the so-called "efficiency" of each link, which is the number of unique queries over the total number of queries received from that link. Nodes preferentially drop links with low efficiency. Thus, in current peer-to-peer networks, nodes use local measurements to improve the performance of the network. Our scheme fits in the same framework and uses locally collected measurements to improve the performance of searching.

There are many other ways to define edge criticality. In a long sequence of theory papers, information concerning "edge criticality" has given novel approximation algorithms for NP-complete problems. In particular, [7] show that there are labels that can be assigned to edges, so that edges across bad cuts of the graph get heavier weights, and doing region growing (weighted breadth first search) finds sparse cuts [8]. Furthermore, [9], [10] show that these labels can be approximated by (repeated computations of) congestion under shortest path routings. Our notion of edge criticality is reminiscent of these techniques.

Another approach to define edge criticality is by considering the principal eigenvectors of the stochastic normalization of the connectivity matrix of the network topology [22]–[25]. Some of these quantities can be also computed efficiently in a distributed setting [23]. It would be very interesting (also, rather hard) to obtain analytical results for heuristics such as the ones proposed in this section.

VI. Experimental Evaluation

In this section, we study experimentally the performance of the hybrid and generalized searching schemes, and the benefits of 1-step replication and lookahead. The experimental results validate the analytical results of the previous sections, and quantify in more detail the performance of the proposed heuristics.

In Section VI-A we present our experimental methodology. In Section VI-B we compare normalized flooding to regular flooding. In Sections VI-C and VI-D we study 1-step replication and shallow lookaheads respectively. In Section VI-E we study the generalized search scheme.

A. Methodology

1) Performance Metrics: In our experiments we are interested in characterizing the performance of searching. We choose some distinct random nodes and perform searching starting from these nodes with the algorithms described in Sections III, IV, and V (typically we use 500 nodes). We measure the number of distinct peers visited per searching per node, which we call *hits*. Our definition of hits relates directly to the standard definition of the number of copies of a specific object discovered by searching, assuming that the

copies of the requested information are placed at random in the network. We also measure the number of propagated messages, which directly relates to the load injected in the network for searching. In addition, we measure the response time of the searching, i.e. the maximum time it takes for the query to complete. More specifically we use the following metrics:
Median and Mean number of distinct peers discovered (hits). Searching algorithms should maximize the median and mean number of distinct peers. The median is a more robust metric, since, in topologies with large irregularities in the degrees, it is possible to measure relatively large mean values because few searches may reach a very large number of users and increase the mean value.
Minimum, Maximum, and Standard Deviation of the number of hits. A large minimum value is important in order to guarantee that the algorithm will have a good worst case performance. The range between the minimum and the maximum values relates to the variation of the performance of the algorithm. The variation is measured using the standard deviation. We believe that algorithms with larger minimum values and smaller variation are preferable.
Number of messages. Good searching schemes strive to minimize the number of messages used to discover as much information as possible. In order to perform a fair comparison of the different searching algorithms we require that they use the same number of messages. Since it is difficult to configure the parameters of each algorithm to guarantee the exact same number of messages, we require that the expected number of messages used in each experiment is approximately the same for all algorithms.
Granularity of number of messages. This is a qualitatively and not quantitative metric. In flooding based algorithms it is difficult to control the parameters of the algorithm, usually the time-to-live, to use a pre-specified number of messages for searching. Linear increases in the TTL result usually in exponential increases in the number of messages. Algorithms with finer granularity are preferable since the user can control the number of messages to reach an adequate number of users (hoping to find enough copies of an item), but, still, not flood the entire network.
Response time. We also measure the maximum running time of each algorithm. In this study we assume a very simple discrete-time model. Each node receives queries from its neighbors and at the same time processes them and forwards copies of the queries, if necessary, to its neighbors. The latter queries will be received at the next unit of time. For all our schemes it is easy to compute the running time of the algorithm, or an upper bound of it. For example, the searching time for flooding with TTL τ is τ. Despite the fact that we do not model many important parameters that affect the searching time, like for example propagation and queuing delays, we believe that our definition of running time can be used to judge the relative performance of the different algorithms.

2) Topologies: We are interested in studying the performance of the searching algorithms in networks with irregularities in the node degrees and in networks with clustered node topologies. Both cases are typical in complex and unstructured communication networks. In peer-to-peer networks, users with very good network connectivity may decide to serve as supernodes. Typically, these users have a much larger number of neighbors; in the Gnutella network for example the average user has 4-6 network connections, whereas a supernode may connect to 20-30 other peers and in many cases have even more neighbors. Also, a common pattern that appears in every unstructured communication network is the clusterness of the topology. The existence of these clusters has been shown to greatly affect the performance of various communication functions in these networks, including searching [1], [25]. There are other parameters that may affect the performance of searching, including the dynamic nature of the topologies, that have been studied elsewhere and which we ignore in this study.

We will use the following synthetic topologies to compare the searching algorithms. Currently, there are limited real data for operational peer-to-peer networks mainly due to the difficulty in collecting such topological information.
Random d-regular Graphs. Extensive analytical work has shown that random d-regular graphs have good properties, including low diameter, good connectivity (i.e. there are no clusters of nodes), small second eigenvalue, and good conductance. We will use d-regular random graphs as a canonical model of a well connected network. Topologies generated with that model will serve as baseline for comparisons. Moreover, the topologies of third generation peer-to-peer networks, like BitTorrent, that use a centralized server to control how the topology is formed, may be more accurately modeled by well connected topologies.
Power Law Graphs. Many seemingly complex networks, including the Internet at the AS level, the Web Graph, and many others, have been shown to be characterized by power-laws. Most typically the degrees of the nodes of the network follow a power-law. The power-law is usually characterized by a parameter λ called the powerlaw exponent, which relates to the probability of observing nodes of high degree according to the formula $\Pr\left[\text{degree} > x\right] \sim 1/x^{\lambda}$. We use topologies with $\lambda = 1.4, 2.0, 2.4, 3.0$. [4], [5] characterize the performance of look ahead for power-law topologies. For $\lambda = 3.0$ the largest degree in 1M nodes graph is less than 100, which brings the parameters close to real peer-to-peer networks.
Bimodal topologies. Along the lines of Section II we assume that there are two types of nodes in the network. Few nodes are connected to a large number of other nodes and, thus, they are very important for the operation of the network. Such nodes, which are typically called ultra-peers, are connected to the Internet through high-speed links and thus they can afford to connect to have many neighbors. The rest of the users have few neighbors. In our experiments in a 250K node graph we have used 500 nodes of degree 500.
Clustered topologies. We assume that there are clusters of users with very good connectivity inside each cluster. The number of links between clusters are limited. In particular, in most of the experiments, we assume that the network is

composed of a small number of clusters, and each cluster is a 3-regular random graph. Typical values are networks of 20 clusters of 10K nodes each, and 100 random connections between each pair of clusters.

B. Normalized Flooding

In this section we verify the results of Section III and compare the performance of normalized flooding to standard flooding. Both schemes perform similarly in regular topologies. However, when the topology contains nodes with high degrees, which is common in large unstructured communication networks, the normalization allows flooding to scale better.

With normalization it is easier to control how many nodes will be reached by the flooding, thus, allowing, searching at a finer granularity (but still the number of nodes increases exponential with the initial TTL). In Table I-A we give the mean number of unique peers discovered as a function of the initial time-to-live for four topologies of 250K nodes. We observe that in regular topologies standard flooding and normalized flooding behave similarly. In the other topologies however, which contain nodes of high degree, the increase in the number of peers is very fast (verifying Theorem 3.2); after the search reaches a high degree node, increasing the TTL by 1 or 2 will result in discovering a large part of the network. This tremendous increase, however, comes at the cost of reduced efficiency in the searching process (see Table I-B). A large number of messages reach already discovered nodes. On the other hand, with normalized flooding (Theorem 3.3) we have more control on the number of messages generated and, moreover, higher searching efficiency. Indeed, even in topologies with nodes of high degree, normalized flooding behaves similarly to searching in regular graphs.

Normalized flooding behaves better than standard flooding with respect to other metrics that are not shown in Table I. For example the standard deviation (normalized by dividing with the mean) is much smaller (up to 4 times) with normalized flooding (observe that the standard deviation is of interest in the cases that the searching has not reached all the nodes of the network for all starting nodes). This indicates that the performance is more concentrated around the mean, and, thus, is less dependent of the starting node. Similar observations exist for other metrics of interest, like the minimum number of nodes discovered.

Moreover, the number of peers discovered as a function of the initial time-to-live follows a more predictable behavior; in contrast, without normalization there is a sharp jump in the number of peers discovered after few high-degree nodes are discovered. In Figure 2 we plot the number of nodes visited by the flooding as a function of the initial time-to-live. The straight line in the case of normalized flooding indicates that the horizon of the search increases exponentially. When searching in regular graphs we observe similar behavior. With flooding without normalization, the increase in topologies with nodes of high degrees is faster than exponential.

Observe that in order to discover the same number of nodes with normalized flooding as with standard flooding, we need

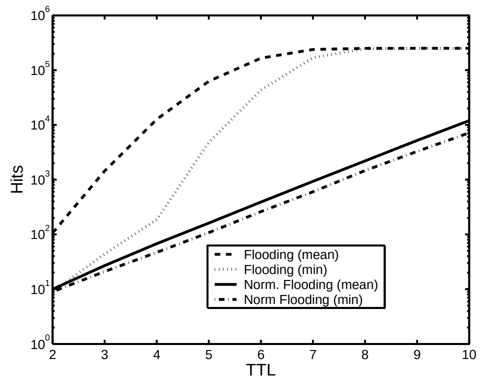

Fig. 2. Number of unique peers discovered as a function of the initial time-to-live. In the case of normalized flooding the number of unique peers increases exponentially with the TTL, and, moreover, the increase is predictable and consistent for all nodes. In the case of regular flooding, the increase is much faster and depends on the node that initiates the flooding.

to increase the TTL. Since the increase of the horizon is more predictable and can be roughly computed by either knowing the properties of the topology or by measuring the increase in the horizon when the TTL increases, it is easy to configure the value of TTL in order to reach at least a pre-specified number of nodes. Despite the fact that the increase in the horizon and in the messages is more predictable, it is still exponential. Later, we describe searching methods that allow finer granularity in controlling the number of messages.

C. Evaluation of 1-step replication

Replication of one step is a practical way to improve the performance of searching by allowing each node to answer queries on behalf of its neighbors. This advantage comes at the cost of replicating information about the content of the neighbors, but, this cost is paid once when a new neighbor arrives and is amortized over a large number of messages that go through the node. This scheme is particularly effective when there are nodes of high degree in the network, as explained in Section IV. Visiting the nodes of large degree, and at the same time their neighbors, guarantees that a large portion of the network has been covered.

The performance of searching in networks of 250K nodes with normalized flooding and with random walks both with and without 1-step replication is given in Table II. This table validates Theorems 4.2 and 4.3. In Table II we report the average number of unique peers discovered, but similar observations are given for the other statistics, like the standard deviation and the minimum number of peers discovered.

Compare Table II with Table I. In the case of regular graphs normalized flooding with 1-step replication and TTL τ behaves very similarly to normalized flooding with TTL $\tau + 1$. In regular graphs random walks with 1-step replication perform better than floodings (in the worst case 20-30%).

The advantage of 1-step replication becomes clear in graphs with large degrees, like the power-law graphs and the bimodal graph of Table II. The number of hits significantly improved

TABLE I

PERFORMANCE AND EFFICIENCY OF FLOODING AND NORMALIZED FLOODING.

A. Average number of unique peers discovered (F: Flooding, NF: Normalized Flooding).

TTL	Regular		Bimodal		Power-Law $\lambda = 1.4$		Power-Law $\lambda = 2.2$	
	F	NF	F	NF	F	NF	F	NF
2	9	9	370.3	9.8	28,216.5	10.9	106.1	10.0
3	21	21	28,463.5	24.9	177,192.9	32.1	1,441.9	27.0
4	45	45	126,581.9	58.5	247,700.5	87.4	12,633.6	67.7
5	93	93	212,759.4	133.6	249,993.2	231.9	62,480.5	161.4
6	188.9	188.9	244,392.6	297.2	249,999	610.6	165,543.6	389.6
7	380.7	380.7	249,635.0	661.7	249,999	1,594.1	238,522.0	930.1
8	763.9	763.9	249,991.4	1,422.8	249,999	4,122.4	249,807.9	2,197.1
9	1,528.7	1,528.7	249,999	3,014.8	249,999	10,430.9	249,999	5,194.6
10	3,051.0	3,051.0	249,999	6,212.4	249,999	24,599.3	249,999	11,953.9

B. Average efficiency of searching (unique peers over messages).

TTL	Regular		Bimodal		Power-Law $\tau = 1.4$		Power-Law $\tau = 2.2$	
	F	NF	F	NF	F	NF	F	NF
2	1.00	1.00	1.00	1.00	0.97	0.99	1.00	1.00
3	1.00	1.00	0.84	1.00	0.42	0.99	1.00	1.00
4	1.00	1.00	0.63	1.00	0.13	0.98	0.96	1.00
5	1.00	1.00	0.46	0.99	0.11	0.96	0.82	1.00
6	1.00	1.00	0.37	0.98	0.11	0.94	0.56	1.00
7	1.00	1.00	0.34	0.96	0.11	0.92	0.33	1.00
8	1.00	1.00	0.34	0.93	0.11	0.88	0.25	0.99
9	1.00	1.00	0.34	0.87	0.11	0.82	0.25	0.98
10	1.00	1.00	0.34	0.82	0.11	0.74	0.25	0.96

Note: Observe the more gradual increase in the number of peers discovered and the more gradual and slow decrease of efficiency when normalized flooding is used. This indicates better granularity when using normalized flooding compared to using regular flooding.

TABLE II

PERFORMANCE OF SEARCHING WITH 1-STEP REPLICATION.

A. Normalized Flooding

TTL	Regular	Bimodal	Power-Law 1.4	Power-Law 2.2	Clustered
2	21	1,150.4	30,219.5	162.3	32.9
3	45	2,580.5	56,629.5	419.1	74.3
4	93.0	5,820.5	88,795.1	1,122.1	162.1
5	188.9	12,256.9	112,658.4	2,605.3	345.7
6	380.7	25,594.8	132,352.9	5,918.2	738.6
7	763.9	49,742.8	152,177.2	12,059.1	1,578.9
8	1,528.7	87,841.0	176,396.6	23,414.3	3,211.6
9	3,051.0	125,520.9	202,415.0	43,747.4	6,329.7
10	6,068.1	146,710.3	224,319.0	79,205.6	11,962.3
Pre-processing	750,000	990,852	2,446,674	1,265,536	1,019,500

B. Random Walks

TTL	Regular	Bimodal	Power-Law 1.4	Power-Law 2.2	Clustered
2	30.1	813.0	26,649.7	133.4	25.2
3	69.1	1,905.0	55,776.0	407.5	55.3
4	148.2	4,030.1	91,807.4	972.3	127.4
5	304.1	8,730.9	113,002.1	2,190.5	270.5
6	616.3	18,045.4	131,526.0	5,214.3	614.8
7	1,239.6	36,256.0	151,263.1	10,746.3	1,347.0
8	2,484.6	67,627.9	174,907.1	20,442.5	2,961.0
9	4,948.0	108,043.0	199,615.3	38,695.5	6,388.4
10	9,806.2	139,712.5	222,433.2	69,877.1	13,678.1
Pre-processing	750,000	990,852	2,446,674	1,265,536	1,019,500

Note: (1) In the case of Random Walks (Table B) the TTL column indicates that the random walker is using the same number of messages as if performing normalized flooding with that TTL. (2) Comparing to Table I, we observe that 1-step replication increased the performance of searching via normalized flooding substantially for the same number of messages.

TABLE III

PERFORMANCE TOPOLOGIES OF 1M NODES.

A. Power-law graph with $\alpha = 2.0$.

Metric	Flooding	Random-Walk	RW-look ahead $\tau = 2$
Median	12,652.50	24,692.50	26,097.00
Mean	27,773.31	24,694.32	28,616.97
Min	281	24,457	13,859
Max	393,040	24,930	71,167
Std	42,876.55	87.17	10,605.26
20-th perc.	996	24,495	14,972
Messages	32,640	32,640	32,851
TTL	4	-	2
Time	4	32,640	430

B. 6-Regular random graph.

Metric	Flooding	Random-Walk	RW-look ahead $\tau = 2$
Median	17,668.50	17,620.00	16,285.50
Mean	17,395.52	17,620.13	16,282.10
Min	10,576	17,389	15,519
Max	19,989	17,816	17,098
Std	1,477.91	73.48	296.20
20-th perc	13,238	17,453	15,667
Messages	22,386	22,386	22,656
TTL	6	-	2
Time	6	22,386	2,152

Note: In the RW-look ahead method, the number of samples between successive floodings is 4.

compared to not using 1-step replication (Table I). The main reason is that both methods, flooding and random walks, quickly discover the nodes of high degree and through them discover a large portion of the nodes in the network. In the cases of graphs with large degrees the performance of normalized flooding is better than random walk (in the worst case by approximately 20%); on the other hand, in topologies with clusters, random walks perform slightly better.

In all cases the preprocessing to implement 1-step replication is given in the last line of Table II. This cost is amortized over a large number of queries.

D. Evaluation of random walk with lookahead

An extension of the previous scheme is to perform a short random walk with shallow local floodings, of TTL $\tau = 2$ on every step (or, every few steps). We call this random walk with lookahead. Since we do not think that it is realistic to maintain replicas of your neighbors' neighbors, therefore we charge the algorithm for all the messages generated by both the random walk and the local flooding. In Table III we show the performance of random walk with lookahead $\tau = 2$ in a regular graph and in power-law graph with few large degrees. The main observation is that the performance of the random walk with lookahead is similar in terms of unique peers discovered to performing a long random walk without lookahead, but, the response time is much smaller. (See also [19] for a different application of lookahead 2.)

E. Edge criticality and searching with weights

In Table IV we give some statistics of the performance of generalized searching in topologies with clusters. This is experimental validation of Section V. We study the performance of generalized searching for different methods of assigning the weights on the edges. The weights depend on the number of the shortest hop paths that go through each link. Recall that we do not compute all the shortest paths, but, instead, sample by computing shortest-paths for a subset of the nodes (2% of the nodes). Since it is possible that the sampling does not cover all edges, we assign a value of one to each uncovered edge. The relative weight that a node assigns to its incident edges depends on the number of shortest-hop paths that use these edges. Assuming that r is this ratio, then, in Table IV, we experiment with different assignments of the weights that are proportional to various powers of that ratio r^i. By increasing the power i, we increase the priority we give to the critical edges, and, as indicated in the table, the performance of searching increases. Increasing the power biases the searching towards the direction that leads to the boundary nodes and, subsequently, to other clusters. Of course, increasing the power i beyond a certain point reduces the performance (the process degenerates into oscillations between clusters).

In regular graphs without clustering, generalized searching performs similarly to standard flooding.

Does generalized searching perform well to other topologies with good connectivity, like for example power-law graphs and bimodal topologies? In other words, could the assignment of edge criticality ever become harmful? In power-law graphs and bimodal topologies edges incident to large degree vertices carry a lot of shortest paths and, hence, are assigned large criticality. However these edges do not belong to bad cuts. Therefore, in non-regular topologies we normalize the edge criticality by dividing the number of paths going through an edge (u, v) by the maximum of the degrees of u and v. We have observed experimentally that using normalized edge criticality makes generalized flooding behave very similarly to flooding. In other words, if we are careful about the normalization, generalized searching does not decrease the performance of known algorithms.

VII. Summary and Further Directions

We studied the performance of various search algorithms in unstructured P2P networks. We quantify performance in terms of number of distinct nodes discovered, the number of propagated messages (network overhead), and the maximum response time. We quantified the performance of flooding in regular P2P networks. We quantified the performance of normalized flooding in non regular P2P networks, and showed that normalization rectifies the problems caused by non regularity. We showed that 1-step replication is helpful in search by random walk as well as search by normalized flooding, especially when the network has a small number of supernodes. We studied hybrid and generalized search schemes that can be viewed as short random walks with shallow local floodings, or floodings with directional information. Our analysis used the theory of random graphs, and our new proposed algorithms are based on edge criticality heuristics previously used in the theory of approximation algorithms. In all cases we have validated our results with extensive experiments in very large topologies.

The random walk method for searching in peer-to-peer networks has been studied in [1], [3]. Hybrid search schemes and networks with supernodes have been discussed in [4]–[6]. Very recently further hybrid search schemes related to percolation theory are discussed in [26].

Generalized searching poses new theoretic questions and we propose analytically quantifying directional flooding as an interesting open problem. Directional flooding is a generic search scheme and we believe it may become of use in different networking contexts, such as sensor and ad-hoc networks. To implement generalized searching we also need practical heuristics to assign edge weights that use little, and preferable passive, network measurements.

Acknowledgments

This work is funded by NSF ITR-0220343.

References

[1] Christos Gkantsidis, Milena Mihail, and Amin Saberi, "Random walks in peer-to-peer networks," in *IEEE Infocom*, Hong Kong, 2004.

[2] Jordan Ritter, "Why gnutella can't scale. no, really.," `http://www.darkridge.com/~jpr5/doc/gnutella.html`, 2001.

[3] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker, "Search and replication in unstructured peer-to-peer networks," in *International Conference on Supercomputing*, New York, New York, USA, 2002, pp. 84–95, ACM Press, Extended version in `http://www.cs.princeton.edu/~qlv/download/searchp2p_full.pdf`.

[4] Lada A. Adamic, Rajan M. Lukose, Bernardo Huberman, and Amit R. Puniyani, "Search in power-law networks," *Physical Review E.*, vol. 64, no. 046135, 2001.

[5] Milena Mihail, Amin Saberi, and Prasad Tetali, "Random walks with lookahead in power law random graphs," Available at `http://www.cc.gatech.edu/fac/Milena.Mihail/lookahead.pdf`, 2004.

[6] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker, "Making gnutella-like networks scalable," in *SIGCOMM*, Karlsruhe, Germany, 2003, pp. 407–418, ACM.

[7] F. T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multicommodity flow problem with applications to approximation algorithms," in *29th IEEE Symp. on Foundations of Computer Science (FOCS'88)*. 1988, pp. 422–431, IEEE.

[8] Vijay V. Vazirani, *Approximation algorithms*, Springer, Berlin; New York, 2001, Vijay V. Vazirani. ill.; 25 cm.

[9] P. Klein, C. Stein, and . Tardos, "Leighton-rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities," in *Annual ACM Symposium on Theory of Computing (STOC)*, Baltimore, Maryland, United States, 1990, pp. 310–321.

[10] Philip Klein, Serge Plotkin, Clifford Stein, and va Tardos, "Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts," *Journal on Computing*, vol. 23, no. 3, pp. 466–487, 1994.

[11] Bella Bollobas and Fernandez de la Vega, "The diameter of random regular graphs," *Combinatorica*, vol. 2, no. 2, pp. 125–134, 1982.

[12] Alan M. Frieze, "Disjoint paths in expander graphs via random walks: A short survey," in *Random '98*. 1998, Lecture Nodes in Computer Science (1518), pp. 1–14, Springer.

[13] Joel H. Spencer, *Ten lectures on the probabilistic method*, CBMS-NSF regional conference series in applied mathematics; 64. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2nd edition, 1994, Joel Spencer. ill.; 26 cm.

[14] "Gnutella," `http://p2pjournal.com/main/gnutella.htm`.

[15] Colin Cooper and Alan Frieze, "The cover time of random regular graphs," Available at `http://www.math.cmu.edu/~af1p/Cover.ps`, 2004.

[16] Colin Cooper and Alan Frieze, "The cover time of sparse random graphs," in *Symposium on Discrete Algorithms (SODA)*, Baltimore, Maryland, 2003, pp. 140 – 147, SIAM/ACM.

[17] R. Impagliazzo and D. Zuckerman, "How to recycle random bits," in *30th IEEE Symposium on the Foundations of Computer Science*, Research Triangle Park, NC, 1989, pp. 248–253, IEEE.

[18] D. Gillman, "A chernoff bound for random walks on expander graphs," *Journal on Computing*, vol. 27, no. 4, pp. 1203–1220, 1998.

[19] Gurmeet Singh Manku, Moni Naor, and Udi Wieder, "Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks," in *ACM Symposium on Theory of Computing (STOC)*, Chicago, IL, USA, 2004, pp. 54 – 63, ACM.

TABLE IV

PERFORMANCE OF GENERALIZED SEARCHING FOR VARIOUS ASSIGNMENTS OF EDGE CRITICALITY.

Method	Min	Mean	Median	Max	Std	Mean Response Time
Proportional	12,869	17,276.30	17,236.0	21,523	1,453.9	8.1
Quadratic	16,054	21,036.31	21,194.0	23,143	1,061.6	10.9
Cubic	18,236	22,248.79	22,386.0	23,292	623.8	14.0
Fifth power	20,037	22,398.52	22,453.0	22,997	296.1	20.7
Tenth power	19,581	21,494.50	21,515.5	22,049	236.8	36.7
Flooding with TTL 6	10,410	16,377.83	15,595.5	30,859	3,223.2	6
Random Walk	20,034	20,328.72	20,330.0	20,591	84.8	28,026
Uniform Sampling	24,252	24,439.99	24.441.5	24,615	50.4	1

Note: The average number of messages in all cases is equal to 28,026. Two links l_1 and l_2 receive budget proportional to the ratio of the number of shortest hop paths going through them in the case of proportional sharing. In the cases of quadratic, cubic, and tenth power the allocation of the budget is proportional to the second, the third, and the tenth power of the ratio respectively.

[20] Christos Gkantsidis, Milena Mihail, and Amin Saberi, “Throughput and congestion in power law graphs,” in *ACM SigMetrics*, San Diego, CA, US, 2003.

[21] “Mutella,” `http://mutella.sourceforge.net/`, 2004.

[22] Fan R. K. Chung, *Spectral graph theory*, Regional conference series in mathematics, no. 92. Published for the Conference Board of the mathematical sciences by the American Mathematical Society, Providence, R.I., 1997, Fan R.K. Chung. 26 cm. ”CBMS Conference on Recent Advances in Spectral Graph Theory held at California State University at Fresno, June 6-10, 1994”–T.p. verso.

[23] David Kempe and Frank McSherry, “A decentralized algorithm for spectral analysis,” in *Proc. the 36th annual ACM symposium on Theory of computing*, Chicago, IL, USA, 2004, pp. 561 – 568, ACM.

[24] Stephen Boyd, Persi Diaconis, and Lin Xiao, “Fastest mixing markov chain on a graph,” *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[25] Christos Gkantsidis, Milena Mihail, and Ellen Zegura, “Spectral analysis of internet topologies,” in *IEEE Infocom*, San Francisco, CA, US, 2003.

[26] N. Sarshar, V.P. Roychowdury, and P. Oscar Boykin, “Percolation-based search on unstructured peer-to-peer networks,” in *4th IEEE International Conference on Peer-to-Peer Computing*, Zurich, Switzerland, 2004.