Decentralized Search in Scale-Free P2P Networks

Praphul Chandra¹ and Dushyant Arora²
HP Labs, Bangalore, India
praphul.chandra@hp.com, dushyantarora13@gmail.com

Abstract—Search in peer-to-peer networks is a challenging problem due to the absence of any centralized control & the limited information available at each node. When information is available about the overall structure of the network, use of this information can significantly improve the efficiency of decentralized search algorithms. Many peer-to-peer networks have been shown to exhibit power-law degree distribution. We propose two new decentralized search algorithms that can be used for efficient search in networks exhibiting scale-free design. Unlike previous work, our algorithms perform efficient search for a large range of power-law coefficients. Our algorithms are also unique in that they complete decentralized searches efficiently even when the network has disconnected components. As a corollary of this, our algorithms are also more resilient to network failure.

Keywords—Peer-to-peer networks; decentralized search; power-law distribution; scale-free networks

I. INTRODUCTION

Many peer-to-peer (P2P) networks have been shown to exhibit a power-law degree distribution. This is often true for large scale peer-to-peer networks which emerge for content sharing e.g. Gnutella & Freenet [5,6]. A graph $G$, is said to exhibit a power-law degree distribution if the number of nodes with $k$ links is proportional to $k^{-\gamma}$, where $\gamma$ is called the power-law exponent of the graph and has a value greater than 1. When the value of the power-law exponent lies between 2 & 3, the second moment $\langle k^2 \rangle$ i.e. the variance of $k$, tends to infinity and such networks are referred to as scale-free networks. In a network exhibiting power-law degree distribution, most nodes possess a small degree and a few handful of nodes act as hubs with very high degree [1].

In large scale P2P networks, the power-law degree distribution is an emergent feature which offers strong network resilience [3]. Since the degree distribution is heterogeneous, a random failure will very likely strike one of the nodes with low degree. The majority of the nodes in the network have a small degree - these nodes are most often not crucial for the connectivity of the network. Hence, in such networks, the giant connected component (GCC), which is likely to exist, persists for high values of node failure rates in the network. This resilience to random failure is what makes the power-law degree distribution attractive to P2P networks which are often characterized by frequent churn of nodes joining & leaving the system [4].

Various models have been proposed to explain the emergence of power-law degree distribution in networks without a central controller/coordinator. Among the most prominent is the preferential attachment model of Barabási and Albert [2] which builds on the notion that new nodes tend to form links with existing nodes in proportion to the degree of the existing nodes. The preferential attachment model has been able to explain the existence of power-law degree distribution in many self-organizing networks like the WWW, citation networks, collaboration graph of movie actors etc. [2]. Clauset and Moore [14] proposed a network evolution model which tries to explain how the World Wide Web develops a power-law degree distribution via rewiring of edges.

Many P2P content sharing networks, in fact, explicitly incorporate the preferential attachment model of growth by design for e.g. in the Gnutella network every new node joining the system first connects to a handful of known servers. A Gnutella client wishing to join the network must find the IP address of an initial node to connect to by looking up ad hoc lists of popular Gnutella servers. This ad hoc method of growth prompts new nodes to connect preferentially to nodes that are already well connected.

II. RELATED WORK

In this paper, we focus our attention on decentralized search in P2P networks which exhibit power-law degree distribution. In general, decentralized search in P2P networks is a difficult problem due to the absence of a centralized controller and due to the limited information available at each node. Thus, the central challenge is to enable a node to find another node by efficiently routing a message/query without global information about each node’s location being made available at a centralized location. In decentralized search, when a particular node ‘i’ wants to route a message to another node ‘j’, it can use traditional approaches like Breadth First Search, Depth First Search and other broadcast policies [11]. However, such approaches are very expensive in time and/or bandwidth consumed.

As an alternate, search algorithms that exploit information about the overall structure of the network have been proposed in earlier work. Adamic et al [1] proposed a decentralized algorithm for search in P2P networks which takes advantage of the power-law degree distribution of these networks and performs better than the flooding-based (e.g. breadth-first) search algorithm which is bandwidth intensive and hence not scalable. Using the generating function approach introduced by Newman [9], Adamic et al. showed that the distribution of the number of links the highest degree neighbor has, among $r$ neighbors is given by

$$p(x, r) = r(1 + x)^{1-\gamma}(\gamma - 2)[1-(x + 1)^{2-\gamma}]^{r-1}$$

$$\times(1-n^2/\gamma-1)^{-r}$$

(1)

where $n$ is the number of nodes and $\gamma$ is the power-law exponent of the graph. Using this distribution they calculated

¹Contact Author
²While intern at HP Labs, India
the expected degree of the highest degree neighbor and plot the ratio between the expected degree of the highest degree neighbor and the degree of a node. They showed that the probability of finding an even higher degree node in the neighborhood of a high degree node depends strongly on the power-law exponent of the graph. For $2.0 < \gamma < 2.3$ and small graphs, Adamic et al [1] showed that forwarding the message to the neighbor with the highest degree is a good approach for performing search. But for larger graphs and higher $\gamma$ values this approach performs poorly. Another limitation of their approach is that it is not capable of handling networks with multiple components. In their simulations, Adamic et al [1] extracted the giant connected component from the network graph and analyzed the performance of their algorithm on this component only.

Kleinberg [12] has proposed decentralized search algorithms in networks which exhibit the small world property. A graph $G$, is said to exhibit the small world property if its average path length is poly-logarithmic in $n$, the number of nodes in the network. This approach is basically a greedy algorithm which assumes that the network is embedded in a structured space (like a lattice). Given the lattice location of the target, each node forwards the message to the neighbor which is closest to the target with respect to the $L_1$ distance on the lattice. The author shows that this approach is efficient for certain values of the dimension of the underlying lattice.

An alternate approach to the problem has been to set-up on an overlay network which is defined specifically for making search easier. Distributed Hash Tables are an example of this approach - Lua et al. [10] present a review of related work.

III. OUR APPROACH

In this paper, we propose two new decentralized search algorithms that can be efficiently used to search through power-law networks with disconnected components and wide range of power-law exponents. The algorithms use local information such as the identities and connectedness of a node’s neighbors, and its neighbors’ neighbors, but not the target’s global position. We do not make any assumptions about the network being embedded in a structured space [12]. We demonstrate that our search algorithms scale sub-linearly with the number of nodes in the network and guarantee completion of search query. Thus, our algorithms increase reliability and availability of peer-to-peer networks. As our algorithms use rewiring for search completion they guarantee completion of search query, even though the search time can be $O(n)$ in the worst-case for a network with $n$ nodes.

In our approach, when a node ‘$i$’ wants to route a message to another node ‘$j$’, it forwards the message to the neighbor which has the highest degree. This is similar to the approach proposed by Adamic et al [1]. The key difference in our approach is how we handle the scenario when a search is unsuccessfully terminated using the basic approach of ‘forward to highest degree neighbor’ strategy of Adamic et al.

![Failure Rate Graph](image)

The graph shows that the number of unsuccessful searches is quite frequent. Earlier works have handled these scenarios by terminating the search unsuccessfully and working only on the giant connected component in the graph. Our approach, on the other hand, proposes a rewiring algorithm to handle such scenarios. We show that our algorithm performs better for a wider range of power-law exponents and can handle networks which have multiple components. This is important since real world P2P networks may exhibit disconnected components due to a variety of reasons e.g. frequent entry and egress of nodes, node failure, existence of underlying network limitations (firewalls etc.).

As mentioned, our key contribution is handling scenarios where the basic approach of ‘forward to highest degree neighbor’ would have unsuccessfully terminated. Our approach is that the node, at which the search has terminated unsuccessfully (called the terminal node), breaks one of its outgoing links and forms a new outgoing link. The link-to-break is chosen randomly from one of the outgoing links of the terminal node and the link-to-make is chosen according to the network evolution rule. We use the preferential attachment model in our algorithms to form links during re-wiring.

In our simulations, we start by generating a network with power-law degree distribution and we use this network to perform decentralized search simulations. We select a source & target node at random and then apply our search algorithm for routing the message. Note that the underlying network does contain disconnected components.

During the simulations, each node forwards the message to the node with the highest degree. Search terminates when the message reaches any one of the second neighbors of the target node. If and when the search reaches a node at which it can no longer proceed (because of earlier mentioned reasons), we allow this node to rewire one of its edges. The rewiring strategy is described below. We note that the degree density of
the network is never changed since no new edges are added in
the graph and we rely strictly on rewiring edges.

The link-to-break is chosen randomly from one of the
(outgoing) links of the terminal node; except that this cannot
be the link through which the message was received. The link-
to-make is chosen according to the network evolution rule, in
our case it follows the preferential attachment model i.e. the
probability, \( p(k) \), that this new link would terminate at a node
with degree \( k \) is \( \sim k^{-\gamma} \). The creation of the new edge according
to preferential attachment rule ensures that the underlying
structure of the network is not violated. Empirical support in
favor of the preferential attachment rule as the reason for the
emergence of power law degree distributions is also well
established \([2,8]\) and has guided our choice. We also ensure
that the target node for rewiring is chosen such that it has not
already been visited by the message – in real implementations,
this can be achieved if each message carries with it a list of all
nodes it has visited in the network. We call this strategy as
**Terminal (T) node rewiring** strategy.

We show that this dynamic model improves upon the
original algorithm in two important ways: (a) the routing times
are lower as compared to the algorithm proposed by Adamic et
al \([1]\) for random power-law graphs of higher power-law
exponent. The algorithm always finds the target if it exists,
also search cost (number of steps until approximately the
whole graph is revealed) scales sub-linearly with the size
of the graph and (b) if the network has disconnected
components, the search can still complete and the algorithm in
fact, connects the disconnected components to form a single
 giant component.

In a small extension of the **Terminal (T) node rewiring**
approach, we handle potentially unsuccessful terminations by
using two re-wirings in the graph instead of one. As before,
the terminal node rewrites as described above. In addition, the
original source node of the message rewrites to the terminal
node. We do not expect this rewiring to affect the results of
search algorithm but this latter rewiring may be seen as a
reward (or an incentive) for the effort undertaken by terminal
node to rewire one of their outgoing edges to the next hop
node. Thus the terminal node’s degree remains constant. We
call this strategy as **Source(S)-T + T node rewiring** strategy.

The empirical motivation for the source rewiring to
terminal node of this model is the notion of incentivizing
nodes to rewire their edges to enable path completion for other
nodes. This notion of incentivizes has been proposed in earlier
work on query incentive networks \([13]\) which notes that
successful decentralized routing in social networks is critically
dependent on individual’s decision to participate in the
routing.

**IV. SIMULATION & ANALYSIS**

For carrying out simulations we generate random power-
law graphs with desired exponents and then simulate our
algorithms on them and compare them with algorithms
proposed in earlier work. Figures 1 & 2 show the degree
distribution before \& after 1,000 iterations of our search
algorithms on random power-law graphs with 10,000 nodes.
We do not extract out the GCC but instead work on the entire
graph. We see that the rewiring process does not significantly
affect the power-law exponent of the graph.

![Figure 2](image2.png)

Figure 2. Degree distribution curves for random power-law graphs with
10,000 nodes (a) before \( \gamma = 2.9 \) (b) after \( \gamma = 2.89 \) for T node rewiring
algorithm.

![Figure 3](image3.png)

Figure 3. Degree distribution curves for random power-law graphs with
10,000 nodes (a) before \( \gamma = 2.89 \) (b) after \( \gamma = 2.93 \) for S-T + T node rewiring
algorithm.

The search cost for these two rewiring models is almost the
same and scales sub-linearly with the size of the graph: similar
to the high degree node seeking strategy of Adamic et al. \([1]\).
Fig. 3 shows this graph.

![Figure 4](image4.png)

Figure 4. This graph shows how search time scales sub-linearly with the
graph size for our two new search algorithms. Simulations were carried on
random power-law graphs of varying size and power-law exponent of 3. We
are here not working on the GCC but the entire graph which has disconnected
components. For calculating search cost we perform 100 iterations for each
graph size.
Fig. 4, 5 and 6 compare the cumulative number of nodes found by our proposed algorithms with the High Degree Node Seeking Strategy (HDNSS) of Adamic et al [1]. We perform our simulations on random power-law graphs with exponents 2, 2.5 and 3 respectively. We can see that both our rewiring strategies perform at par with HDNSS for power-law graphs with exponent = 2, a little better than HDNSS for power-law exponent = 2.5 and vastly better than HDNSS for power-law graphs with exponent = 3.

![Cumulative Nodes Found Graph](image1.png)

Figure 5. Cumulative nodes found data for random power-law graph with 10,000 nodes having 164 connected components and power-law exponent of 2 and giant connected component with 9398 nodes. Our algorithms perform at par with HDNSS algorithm.

![Cumulative Nodes Found Graph](image2.png)

Figure 6. Cumulative nodes found data for random power-law graph with 10,000 nodes having 852 connected components and power-law exponent of 2.5 and giant connected component with 8185 nodes. We can see that HDNSS fails to discover new nodes after some steps. Our algorithms allow the message to propagate further by allowing rewiring. Eventually, our algorithms will traverse all nodes in the graph.

In order to test the performance of our algorithms in the presence of disconnected components, we perform another simulation on a random power-law graph with 10,000 nodes and power-law exponent = 2. If the source and target are in separate components then the routing time will be infinite for HDNSS algorithm. Therefore we use a new metric (1/Routing Time) to compare the performance of our algorithms with HDNSS. A better routing algorithm will have lower Routing times and thus higher value of this metric (1/Routing Time). Fig. 7 shows that our algorithms perform better than HDNSS consistently for various graph sizes. The value of the metric will be higher as we increase the power-law exponent of the graph or the graph size as this leads to an increase in the number of connected components in the graph.

![Cumulative Nodes Found Graph](image3.png)

Figure 7. Cumulative nodes found data for random power-law graph with 100,000 nodes having 21,841 connected components and power-law exponent of 3 and giant connected component with 47,085 nodes. Our algorithms perform much better as compared with HDNSS.

![Routing Times In Disconnected Graphs](image4.png)

Figure 8. Routing time graph for random power-law graphs of varying size and power-law exponent of 2. We calculate the average value of 1/Routing Time for 1,000 iterations of each algorithm using random source and target nodes. This process is repeated 10 times for each graph size to determine the final value of the metric for that graph size. Table 1 shows the average number of connected components for each graph size.
TABLE I. THIS TABLE SHOWS THE NUMBER OF CONNECTED COMPONENTS FOR EACH GRAPH SIZE AVERAGED OVER 10 ITERATIONS

<table>
<thead>
<tr>
<th>Size</th>
<th>Average number of connected components</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>11</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>2000</td>
<td>43</td>
</tr>
<tr>
<td>4000</td>
<td>75</td>
</tr>
<tr>
<td>6000</td>
<td>104</td>
</tr>
<tr>
<td>8000</td>
<td>131</td>
</tr>
<tr>
<td>10000</td>
<td>158</td>
</tr>
</tbody>
</table>

a. The average number of connected components increases as the graph size increases.

Next we simulate how our algorithms cope up with network failure. In Fig. 8 we plot a graph between the metric 1/Routing Time and the network failure percentage. We see that as the network failure percentage increases, the number of connected components in the graph also increases and the performance of our algorithms improves as compared to HDNSS. Also in Table 2 we show that our algorithms reduce the number of connected components of the graph because of their inherent rewiring mechanisms. Thus, our algorithms increase the overall reliability and availability of the network. Also they help in recovery of the network after node failure. For this simulation we first plot a random power-law graph of 10,000 nodes with exponent = 2. We extract the giant connected component from this graph. Next we simulate network failure by randomly removing nodes from the network. This leads to creation of disconnected components. Now we perform 1,000 search iterations of HDNSS, T node rewiring and S-T + T node rewiring algorithms using random source and target nodes.

![Routing Time Vs Failure Rate Graph](image)

Figure 9. Routing time Vs Failure rate graph for random power-law network with 10,000 nodes and power-law exponent of 2. We can see that our rewiring algorithms are more resilient to failure while the performance of HDNSS keeps deteriorating as the failure rate increases. Also our rewiring algorithms reduce the number of disconnected components in the graph. See Table 2.

TABLE II. THIS TABLE SHOWS HOW OUR SEARCH ALGORITHMS HELP IN RECOVERY OF THE NETWORK AFTER NODE FAILURE

<table>
<thead>
<tr>
<th>Failure Rate</th>
<th>Number of connected components after network failure</th>
<th>Number of connected components after T node rewiring</th>
<th>Number of connected components after S-T + T node rewiring</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>48</td>
<td>43</td>
<td>37</td>
</tr>
<tr>
<td>2.5%</td>
<td>115</td>
<td>105</td>
<td>106</td>
</tr>
<tr>
<td>5%</td>
<td>187</td>
<td>161</td>
<td>161</td>
</tr>
<tr>
<td>10%</td>
<td>392</td>
<td>328</td>
<td>320</td>
</tr>
<tr>
<td>20%</td>
<td>804</td>
<td>660</td>
<td>666</td>
</tr>
<tr>
<td>30%</td>
<td>1387</td>
<td>1141</td>
<td>1124</td>
</tr>
<tr>
<td>40%</td>
<td>1646</td>
<td>1285</td>
<td>1298</td>
</tr>
<tr>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. The number of connected components reduces after performing search using our algorithms.

V. CONCLUSION

In flooding-based breadth-first algorithms there is a lot of wastage of network bandwidth as each nodes forwards the message to all its neighbors instead of capitalizing on the power-law structure of the network. Adamic et al. [1] proposed a new decentralized forwarding which scales sub-linearly with the network size. Each node now passes the message to its neighbor of highest degree instead of flooding it to all its neighbors. But this algorithm performs poorly in case of graphs with disconnected components and graphs with power-law exponent higher than 2.3. We propose two new rewiring algorithms that help alleviate these problems. Our algorithms also scale sub-linearly with the size of the graph and also work if the source and target are in different connected components. Our algorithms guarantee search completion if the target exists in the network. Also if there is a partial failure in the network our algorithms cope up much better as compared to HDNSS and also connect the disconnected components along with search completion.

Our rewiring strategy has applications in a number of power-law networks. Of particular interest are the peer-to-peer content sharing networks since rewiring in such networks has very low cost and the primary interest of participating nodes is query completion. There is also the concept of inherent cooperation in these networks and the network evolution model is Preferential Attachment [1] i.e. nodes will preferentially attach to nodes that are hubs of the network. These networks are dynamic in nature i.e. nodes join and leave the network frequently which can lead to creation of disconnected components in the network. The S-T + T node rewiring which has inherent feature of a query incentives and cooperation model is very aptly applicable to the peer-to-peer overlay networks.

The implementation of our algorithm requires the participating nodes of the GNUTELLA network to keep the lists of the files stored by their first and second neighbors. This information must be updated periodically depending on the typical lifetime of nodes in the network. Also since we avoid revisiting a node while performing the search algorithm, we need to append the IP addresses of the nodes already queried to the search message.
REFERENCES


