Analysis of Peer-to-Peer Systems: A Graph-Theoretic Answer to the Diameter-Degree Tradeoff

Dmitri Loguinov, Member, IEEE, Anuj Kumar, Vivek Rai, Sai Ganesh

Abstract – This paper examines graph-theoretic properties of existing peer-to-peer architectures and proposes a new infrastructure based on optimal-diameter de Bruijn graphs. Since rapid expanders exemplified by the class of generalized de Bruijn graphs possess very small routing diameter and very high resilience to node failure, they are well suited for peer-to-peer networks. Using the example of Chord, CAN, and de Bruijn, we first study average routable distances and clustering properties (i.e., “small-world” phenomena) of each graph. We then examine the Laplacian spectrum, algebraic connectivity, path overlap, and several other graph properties that affect routing and resilience of peer-to-peer networks. Having confirmed that de Bruijn graphs offer the best routing distance and highest connectivity among the existing peer-to-peer structures, we offer an incremental building process that preserves optimal properties of de Bruijn graphs under uniform user joins. We call the combined peer-to-peer architecture ODRI – Optimal Diameter and Resilience Infrastructure.

I. INTRODUCTION

In the last few years, peer-to-peer networks have rapidly evolved and have become an important part of the existing Internet culture. All current peer-to-peer proposals are built using application-layer overlay structures, each with a set of graph-theoretic properties that determine its routing efficiency and resilience to node failure. Graphs in peer-to-peer networks range from star-like trees (centralized approaches such as Napster [30]) to complex k-node-connected graphs (such as Chord [42], CAN [33] and Pastry [36]). The performance of each peer-to-peer architecture is determined by the properties of these graphs, which typically possess $\Theta(\log N)$ diameter and $\Theta(\log N)$ degree at each node (where $N$ is the number of peers in the system). Understanding whether these bounds are optimal and whether there exist fixed-degree graphs that can route in $\log N$ time is believed to be the fundamental question of current DHT research [34], [45].

Besides the diameter and degree, a third very important property of a peer-to-peer structure is its resilience to simultaneous node failure. Without fault resilience, the answer to logarithmic routing in fixed-degree networks is obvious and includes a variety of simple tree-like structures. The main goal of this work is to find a fixed-degree graph with provably (not just asymptotically) minimum diameter for a given number of nodes and maximum connectivity (i.e., fault resilience) among all such graphs.

Another dimension to this work is to provide a unifying analytical framework for understanding the various properties of existing and future peer-to-peer graphs. Many of the existing proposals (e.g., CAN, Chord, Pastry) have not been modeled in ways that can provide a clear quantitative assessment of each graph’s resilience to node failure. In addition to classical approaches, there are proposals entirely based on heuristics (e.g., [18], [28], [43]) with no underlying foundation for choosing one or another graph structure. Our work supplements such proposals with a more fundamental insight into the problem and offers analytical tools for evaluating future peer-to-peer routing architectures.

The paper is organized as follows. We first examine the problem of obtaining logarithmic routing diameter in fixed-degree (sparse) graphs and show how such graphs can be applied in real peer-to-peer networks. Our work relies on generalized de Bruijn graphs [8], [23] of fixed degree $k$ and diameter $\log N$. However, since the diameter itself does not tell the whole story, we also study the average distances between all pairs of nodes since this metric (rather than the diameter) is more likely to affect an average user.

We next examine clustering and small-world properties of each graph and explain how they relate to the expansion properties of the graph. We derive that de Bruijn graphs have an order of magnitude smaller clustering coefficients than Chord, which explains the differences in expansion, resilience, and diameter between the two graphs. We also find a simple graph-theoretic definition of the clustering coefficient based on cycles found in the undirected version of each graph and show its relationship to the isoperimetric number of the graph.

We further show that fixed-degree graphs with logarithmic routing alone are not sufficient for a successful peer-to-peer architecture. Another very important property of a peer-to-peer graph is its resilience to node failure (or simply its connectivity). Connectivity determines how many failures (and where) a graph can tolerate without becoming disconnected. We use the Laplacian spectrum of each graph to show that de Bruijn graphs possess a significantly higher algebraic connectivity than any of the other graphs under study. In the spatial domain, we also examine the amount of overlap between alternative (parallel) paths leading to any given destination and its relationship to the clustering coefficient of each graph.

Having confirmed that de Bruijn graphs offer a resilient and diameter-optimal routing structure, we provide an algorithm for building such graphs incrementally as peer nodes join and leave the system. We also prove that under randomized joins, the node degree at the application layer remains

All authors are with Texas A&M University, College Station, TX 77843 USA ({dmitri, anujk, vivekr, ssai}@cs.tamu.edu, 979-845-0512).
fixed and the application-layer graph is also a de Bruijn graph with high probability.

Besides providing a novel routing graph structure for peer-to-peer networks, our work offers an extensive graph-theoretic evaluation of existing peer-to-peer architectures, which in itself is a major contribution of this paper.

II. BACKGROUND

Many current peer-to-peer networks [33], [36], [42], [46] are based on distributed hash tables (DHTs), which provide a decentralized, many-to-one mapping between user objects and peers. This mapping is accomplished by organizing the peers in some virtual coordinate space and hashing each object (or its name) to these virtual coordinates. The information about each object (such as the IP address of the owner) is kept by the peer to whose coordinates the object hashes. Distributed structure of DHTs, short routing distances, flexibility, and failure resilience make them highly suitable for peer-to-peer networks.

A. Peer-to-Peer DHTs

The original work on DHTs by Plaxton et al. [32] supported prefix-based\(^1\) logarithmic-time routing; however, it was not flexible enough to accommodate node join and departure. Prefix-based routing usually employs a vector space constructed of symbols from some alphabet \(\Sigma\) of size \(b\) and greedy (local) decisions based on matching the next digit of the target peer to that of each neighbor. Plaxton’s work [32] is extended by Zhao et al. [46] in Tapestry and later Bayeux [47], [21]. Prefix-based routing in generalized hypercubes was proposed by Rowstron et al. [36] under the name of Pastry (similar hypercube methods can also be found in [39]). Pastry builds a regular graph with diameter \(\log_b N\) and node degree \((b-1)\log_b N, b \geq 2\). Ratnasamy et al. [33] propose a peer-to-peer architecture called Content-Addressable Network (CAN) that maps the DHT to a \(d\)-dimensional Cartesian space. CAN’s diameter is \(1/2d(N^{d-1})\) and the degree of each node is \(2d\). Stoica et al. [42] propose an alternative hash-based distributed architecture called Chord, which uses a 1D modulo field with both the diameter and degree equal to \(\log_b N\).

Recent proposals start to address the issue of routing in logarithmic time in fixed-degree graphs. For example, Considine et al. [12] expand on Chord’s ring structure by building a digraph (directed graph) of degree two; however, routing in this graph is quite suboptimal (from the graph-theoretic point of view) and requires \(4\log_b N\) steps. Furthermore, the proposed structure in [12] needs to estimate the number of active nodes to properly adjust the size of the ring. Among tree-based structures, Freedman et al. [17] propose a DHT based on distributed tries; however, the paper does not examine degree-diameter tradeoffs nor does it address the issue of resilience. Xu et al. [45] study an almost identical problem to ours and propose a graph based on the static butterfly. Another peer-to-peer development based on butterfly networks is shown in [27].

Other recent work (Liben-Nowell et al. [26]) examines error resilience dynamics of Chord when nodes join/leave the system and derives lower bounds on the number of neighbors necessary at each node in order to maintain a connected graph with high probability. Additional studies also link fault-tolerance to the out-degree of each node. Saia et al. [37] extend CAN to create a highly fault-resilient structure; however, their resulting state at each node increases to \(O(\log^3 N)\). Other fault-tolerant structures include Censorship Resistant Networks with \(O(\log^2 N)\) routing time [16].

B. Optimal-Diameter Graphs

The problem of designing an optimal-diameter graph of fixed degree has been extensively studied in the past. In one formulation of this problem, assume a graph of fixed degree \(k\) and diameter \(D\) (the maximum distance between any two nodes in the graph). What is the maximum number of nodes \(N\) that can be packed into any such graph? A well-known result is the Moore bound [7], [9], [22]:

\[
N \leq 1 + k + k^2 + \ldots + k^D = \frac{k^{D+1} - 1}{k-1} = N_M. \tag{1}
\]

Interestingly, Moore bound \(N_M\) is only achievable for trivial values of \(k\) and \(D\). In fact, the Moore bound is provably not achievable for any non-trivial graph [7], [22]. Directed de Bruijn graphs [8] come close to the Moore bound and can be built with \(N = k^D\) nodes [5], [23] or even with \(N = k^D + k^{D-1}\) nodes [35]. In general, it is not known how close we can approach upper bound \(N_M\) for non-trivial graphs [9]. In the context of peer-to-peer DHTs, we are concerned with a different formulation of the problem: given \(N\) nodes and fixed degree \(k\), what is the minimum diameter in any graph built on top of these \(N\) nodes? The answer readily follows from (1):

\[
D \geq \left\lceil \log_k (N(k-1)+1) \right\rceil - 1 = D_M. \tag{2}
\]

Imase et al. [23] construct nearly-optimal de Bruijn graphs of diameter \(D = \left\lceil \log_k N \right\rceil\), which is at most one larger than \(D_M\). In this paper, we use Imase’s algorithms [23] even though they can be slightly improved [35]. De Bruijn graphs have been previously used for inter-processor communications [6], VLSI design [38], and other operating systems and networking issues [14], [20], [40]. See [4] for a thorough overview of various graph structures used in inter-process communications.

C. Random Graphs

Another direction for building DHTs relies on properties of random graphs. The main thrust in this area is to build logarithmic-time routing structures with constant degree. It is shown in [11] that certain families of random graphs (including power-law graphs) of \(N\) nodes with average degree \(k\) are expected to have diameter \(O(\log N)\) with high probability. Thus, Pandurangan et al. [31] propose a random DHT graph with a constant degree and (almost certainly) logarithmic diameter; however, the paper does not provide an efficient routing algorithm for the proposed structure that can deterministically explore the low diameter of the graph. Aspnes et al. [1] further examine random graphs of fixed degree \((l+1)\) and derive upper and lower bounds on expected routing distance in

\(^1\) We use terms “suffix-based” and “prefix-based” interchangeably when referring to routing.
such graphs. Their results show that both bounds are proportional to \(\ln^3 N / (\ln \ln N)\). Law et al. [25] build random 2d-regular expander graph based on Hamiltonian cycles into layers with \(O(\log N)\) diameter and \(O(\log N)\) neighbors.

Even though random graphs of logarithmic diameter can be built with high probability using random neighbor selection, the design of efficient routing algorithms competitive to those in deterministic graphs is still an open issue.

III. BASICS OF DE BRUIJN GRAPHS

A. Motivation

One of the goals of this work is to build a distributed hash table (DHT) on top of fixed-degree graphs with provably optimal routing time. Since non-trivial Moore graphs do not exist [7], [22], we use de Bruijn graphs [23] of diameter \(\lceil \log_2 N \rceil\) (which is no more than \(D_m + 1\)) and often call them “optimal” since among the class of practically achievable graphs, they are very close to being optimal. To illustrate the impressive reduction in diameter compared to existing structures, assume 1 million nodes and degree \(k\) fixed at \(\lceil \log_2 N \rceil = 20\). Under these circumstances, Chord offers a graph with diameter \(D\) equal to \(\lceil \log_2 N \rceil = 20\), while a de Bruijn graph with the same number of neighbors has a diameter four times smaller: \(D = \lceil \log_2 N \rceil = 5\).

Throughout the paper, we are concerned with the properties of the underlying graph of each peer-to-peer network. Consequently, we examine the diameter and resilience of these graphs assuming that the hashing function uniformly spreads users along the DHT space and that all graphs are populated with the maximum number of nodes. Thus, we leave the study of unbalanced and incomplete DHTs as a separate topic for future work since it has little to do with the graph structure itself and largely depends on the user arrival/departure process as well as the hashing function deployed by the peers. We further assume for simplicity of notation that the total number of nodes \(N\) is a power of node degree and omit ceiling functions whenever appropriate.

B. Structure of de Bruijn Graphs

De Bruijn graphs [23] are nearly-optimal, fixed-degree digraphs of diameter \(\log_2 N\), where \(k\) is the fixed degree of each node and \(N\) is the total number of nodes. Note that de Bruijn graphs are directed graphs with \(k\) outgoing and \(k\) incoming edges at each node, which also holds for many other protocols [42], [46], [36]. Assume that each node \(x\) is hashed to string \(H_x\), drawn from some alphabet \(\Sigma\) of size \(k\). For example, if \(k = 2\), then \(H_x\) is a binary string of \(0s\) and \(1s\). A directed de Bruijn graph [23] contains \(N = k^D\) nodes where \(D\) is the diameter of the graph. Each node \(H_i\) in the graph is a string \((h_1, \ldots, h_D)\) of length \(D\) linked to \(k\) other nodes \((h_2, \ldots, h_D, \alpha)\) for all possible \(\alpha \in \Sigma\). A classical de Bruijn graph for \(k = 2\) and \(N = 8\) is shown in Figure 1 (left) [40]. The diameter of the graph in the figure is 3, even though some nodes link to themselves. In fact, de Bruijn graphs contain exactly \(k\) nodes \((h, h, \ldots, h)\), for all \(h \in \Sigma\), which link to themselves (this makes the graph irregular). It is possible to create a regular de Bruijn graph by linking these nodes to each other for added failure resilience, which we discuss in section 6.

C. Routing

Recall that shortest-path routing between any two nodes in de Bruijn networks follows a deterministic procedure executed by individual nodes in a distributed fashion [23], [40]. Assume node \(x\) seeks a shortest path to node \(y\). The choice of the next-hop neighbor follows a simple string-matching algorithm shown in Figure 1 (right). Node \(x\) finds the longest overlap between the suffix of its hash index \(H_x\) and the prefix of \(y\)’s hash index \(H_y\). In the figure, the longest overlap, labeled \(B\), contains \((D-i)\) digits, for some \(i\). By merging prefix \(A\), overlap \(B\), and suffix \(C\), node \(x\) can generate the entire path \(P\) to reach \(y\), which contains all nodes along the shortest path. Notice that \(P\) starts with \(H_x\), ends with \(H_y\), and contains no more than \(D-1\) intermediate nodes (each node is a \(D\)-character substring of \(P\), read from left to right). As an example, again consider the graph in Figure 1 (left). Suppose node 001 needs to route to node 101 along the shortest path. Using the above procedure, prefix \(A\) is (00), overlap \(B\) is (1), and suffix \(C\) is (01). The resulting shortest path is \(P = (00101)\), which translates to (001) \(\rightarrow\) (010) \(\rightarrow\) (101).

D. Comparison with Existing Graphs

In this section, we briefly examine diameter-degree trade-offs of the existing protocols and compare them to those of de Bruijn graphs. We leave a thorough analysis of numerous recently proposed graphs [18], [27], [25], [28], [43], [45] for future work and conduct a detailed study of two classical approaches Chord [42] and CAN [33] in sections 4-6. This section also shows asymptotic results for Pastry (without detailed analysis) and omits Tapestry since the two graphs are very similar in their diameter and degree at each node.\(^2\)

As an illustration of fixed-degree tree structures, we also examine \(k\)-ary tries as they have been recently proposed for DHTs [17], [43]. A \(k\)-ary trie uses prefix-based routing over a tree where each parent maintains \(k\) children, one child for each symbol in the alphabet. Consequently, the maximum degree of any node in this trie is \((k+1)\) and the diameter of the graph is \(2\lceil \log_2 N \rceil\) (i.e., the distance to the root and back).

Table 1 shows asymptotic routing times and node degrees for de Bruijn graphs and several existing (deterministic) structures. First note that we assume (since [33] does not define peering rules analytically) that CAN uses circular routing in each of the dimensions, which means that all nodes along the borders maintain 2d neighbors and that the CAN graph is regular. Second, Chord’s performance can be viewed as a special case of Pastry for \(b = 2\) even though the two structures have little in common. Finally, the trie structure maintains the

\(^2\) Strictly speaking, Tapestry’s routing table size is \(b \log_2 N\) [46].
average degree over all nodes equal to only 2 (since approximately \((k-1)/k\) fraction of the nodes are leaves); however, the imbalance in the middle of the tree with nodes of degree \(k+1\) creates a rather pessimistic diameter-degree tradeoff. It is interesting that the trie has a very small average degree with an asymptotically optimal diameter; however, fault resilience of this structure is minimal since failure of any non-leaf node disconnects the graph.

<table>
<thead>
<tr>
<th></th>
<th>de Bruijn</th>
<th>Trie</th>
<th>Chord [33]</th>
<th>CAN [33]</th>
<th>Pastry [36]</th>
</tr>
</thead>
<tbody>
<tr>
<td>degree</td>
<td>(\log N)</td>
<td>(k+1)</td>
<td>(\log N)</td>
<td>(2d)</td>
<td>((b-1)\log_N N)</td>
</tr>
<tr>
<td>diameter</td>
<td>(\log N)</td>
<td>(2\log N)</td>
<td>(\log N)</td>
<td>(1/2\log N)</td>
<td>(\log N)</td>
</tr>
</tbody>
</table>

Table 1. Asymptotic properties of the different graphs.

We next examine the performance of these graphs in a hypothetical peer-to-peer system of \(N = 10^6\) nodes. In Figure 2, we plot the diameter of each graph as a function of its degree \(k\). Notice that for low-degree networks \((k < 40)\), even the trie offers a better diameter than the three classical approaches (i.e., CAN, Chord, and Pastry). In fact, the trie routes in half the time compared to Chord. Temporarily disregarding fault resilience of each graph, it is obvious that even the simplest tree structures can provide more efficient routing than the existing DHT approaches. However, trees are notoriously susceptible to node failure and become disconnected when any non-leaf node fails. Consequently, we will spend extensive time later in the paper analyzing fault resilience (connectivity) of de Bruijn graphs and comparing it with that of existing schemes.

Figure 2. Diameter-degree tradeoffs of the existing and proposed graphs for \(N = 10^6\).

Examining Figure 2 and Table 2, we notice that de Bruijn graphs with the same number of neighbors offer diameters at least 4 times smaller than those in Chord and CAN. Furthermore, de Bruijn graphs can route between any pair of nodes in 20 hops with only 2 neighbors, which is 10 times less state than that required by the other three methods to achieve the same diameter.

One interesting observation about CAN points to the fact that selection of the number of dimensions \(d\) is a very important decision for a given number of nodes \(N\). It is noted in [33] that \(d\) is likely to be fixed while \(N\) changes; however, as the figure and the table above show, many values of \(d\) result in suboptimal performance. For a given \(N\), CAN achieves its lowest diameter when \(d = \ln N\) (which can be shown by minimizing \(dN^{1/k}\) while other values of \(d\) can lead to substantially larger diameters than the optimal. Thus, for \(N = 10^6\), \(d_{opt} = 14\) (28 neighbors) and \(D_{opt} = 19\), while diameters as high as 1,000 are possible with other choices of \(d\). Also note that when \(d = \log N/2\), CAN’s degree and diameter are both equal to that of Chord (this is shown in Figure 2 where the two curves overlap and also noted in [33]).

<table>
<thead>
<tr>
<th>node degree (k)</th>
<th>de Bruijn</th>
<th>Trie</th>
<th>Chord [42]</th>
<th>CAN [33]</th>
<th>Pastry [36]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>40</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>25</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>12</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>50</td>
<td>4</td>
<td>7</td>
<td>–</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>160</td>
<td>3</td>
<td>6</td>
<td>–</td>
<td>48</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2. Graph diameter for \(N = 10^6\) (cells with a dash indicate that the graph does not support the corresponding node degree).

IV. ROUTING ANALYSIS

De Bruijn graphs have desirable properties for peer-to-peer networks that stem from their small diameter. However, the diameter of a graph is simply the largest distance between any pair of nodes and only provides an upper bound on the delay experienced by the users. A much more balanced metric is the average distance (delay) between any pair of nodes since this is the performance a user can expect from the peer-to-peer system when searching for objects. In fact, it is possible to reduce the diameter of a graph and at the same time increase its average routing distance as recently demonstrated in [45].

Intuitively, it is clear that small-diameter graphs are so densely packed, that there will be very small improvement between the maximum and the average distance in the graph. Below we show this result for de Bruijn graphs and examine similar metrics for several existing structures. Define \(d(x,y)\) to be the shortest distance between any pair of nodes \(x\) and \(y\) in a given graph. To better understand how the distribution of \(d(x,y)\) is formed and study expansion properties of each graph, we first derive the density (mass) function for \(d(x,y)\) and then compute its expectation \(\mu_d\).

A. Chord

Stoica et al. [42] showed in simulation that the average inter-node distance \(\mu_d\) in Chord is \(D/2\) and offered a simple explanation of why this happens. They further showed the distribution of \(d(x,y)\) to be bell-shaped as demonstrated in Figure 3 (left) for \(N = 1,024\). The curve appears to be Gaussian as illustrated by an almost-perfect fit in the figure. It has been noted before that certain real-world graphs (such as those describing webpage linkage structure [2]) exhibit Gaussian distributions for \(d(x,y)\), but no explanation of why this happens has been offered. We analyze this issue below and derive the answer to this question for Chord.
Recall that in Chord, each node \( x \) with hash index \( H_x \) has \( k = \log_2 N \) neighbors located at indexes \((H_x + 2^i) \mod N\), for \( i = 0 \ldots k-1 \). Notice that any shortest path to a given destination is a sequence of \textit{unique} jumps, each of which is a power of 2. The uniqueness of jumps is easy to see since any two jumps of length \( 2^j \) within the same shortest path can be replaced with a single (more optimal) jump of size \( 2^j+1 \). Consequently, any path of length \( n \) is formed by drawing \( n \) unique elements from set \{1, 2, 4, \ldots, 2^{k-1}\}, where \( D \) is the diameter of the graph as discussed earlier. The number of possibilities to draw \( n \) objects from a set of size \( D \) is \( C(D, n) \), which leads to the fact that each node can reach exactly \( C(D, n) \) other nodes within \( n \) hops. Using symmetry of nodes in Chord, the PMF of \( d(x, y) \) is given by a binomial distribution with parameters \( p = q = \frac{1}{2} \) (recall that \( N = 2^k \)):

\[
p(n) = \binom{D}{n} = \frac{1}{N} \cdot \frac{1}{2^{D-n}2^n} = \left(\frac{D}{n}\right) p^{D-n} q^n, \tag{3}
\]

where \( p(n) \) is the PMF of shortest distances \( d(x, y) \). Our simulation results confirm that (3) gives the exact distribution of shortest path lengths in Chord. The expected value \( \mu_d \) of a binomial random variable is a well known result and equals \( Dp \), or simply \( D/2 \). This provides an alternative derivation of the result previously shown in [42].

![Figure 3](distribution.png)

Figure 3. Distribution of shortest paths \( d(x, y) \) in Chord for \( N = 1024 \) together with a Gaussian model (left). Distribution of path lengths in non-circular CAN, for \( N = 10 \) (\( d = 1 \)), \( 10^2 \) (\( d = 2 \)), \( 10^3 \) (\( d = 3 \)), \( 10^4 \) (\( d = 4 \)) nodes (right).

The reason why binomial distributions appear to be Gaussian is explained by the de Moivre-Laplace theorem, which states that the binomial distribution in (3) asymptotically tends to a Gaussian distribution with mean \( \mu_d = D/2 \) and variance \( Dpq = D/4 \) for sufficiently large \( D \). Even though we have not obtained insight into why certain Internet graphs exhibit Gaussian distributions of shortest paths, we found a clear explanation of this phenomenon in Chord.

There is also a simple intuitive link between the bell shape of the curve in Figure 3 (left) and expansion properties of the graph. As the distance from any given node \( x \) increases, the number of \textit{new} neighbors found by the search slowly saturates and starts declining after half the nodes have been reached. This means that many of the newly found nodes link to some of the previously discovered nodes. This leads to a situation when the new neighbors “know” many of the old neighbors, which is often called the \textit{small-world property} (or the \textit{clustering coefficient}) of the graph [2], [3]. In graph theory, the growth in the number of new neighbors discovered at a certain distance is called node expansion of the graph [24], [29]. For example, very fast expander graphs (such as de Bruijn) maintain exponentially increasing number of new neighbors up to the diameter of the graph, which means that very few of the new neighbors “know” the old ones (and hence their clustering coefficients are virtually zero). We study this phenomenon more carefully in section V, but currently note that we should expect reasonably high values of clustering from Chord.

### B. CAN

Recall that CAN organizes its nodes into a \( d \)-dimensional Cartesian space. We start with a very simple derivation for a 1D case and then generalize it to higher dimensions. Assume that all nodes are uniformly distributed along the \( x \)-axis between 0 and \( N-1 \). To understand advantages of circular CAN spaces, we first assume a \textit{non-circular} version of CAN with diameter \( D = N-1 \). The mean distance between any pair of nodes in a non-circular CAN is given by:

\[
\mu_d = \frac{1}{N^2} \sum_{s=0}^{N-1} \sum_{w=0}^{N-1} |s-w| \approx \frac{1}{N^{2-d}} \int_0^1 \int_0^1 |s-w| ds dw, \tag{4}
\]

where \( s \) is the coordinate of each source and \( w \) is the coordinate of each destination. The discrete sum evaluates to \( \mu_d = (N^2-1)/3N \), however, for large \( N \), we can write an asymptotic result \( \mu_d \approx N/3 = (D+1)/3 \), which is also produced by the integral approximation in (4). This derivation is easily generalized to a \( d \)-dimensional coordinate space by computing the distances between all sources \( s = (s_1, \ldots, s_d) \) and all destinations \( w = (w_1, \ldots, w_d) \). Noticing that the sum below is separable and applying the result from (4):

\[
\mu_d = \sum_{s_1} \sum_{s_2} \ldots \sum_{s_d} \sum_{w_1} \sum_{w_2} \ldots \sum_{w_d} |s_1 - w_1| + |s_2 - w_2| + \ldots + |s_d - w_d| \approx \frac{1}{3} \left((N^{2-d} - 1)/D + d - o(1)\right) \approx \frac{D + d}{3}.
\]

For \textit{circular} CAN spaces, the result is slightly different and confirms the one shown without proof in [33]. Since any pair of nodes are within \( D = \lceil (N-1)/2 \rceil \) distance from each other in circular 1D CAN, (4) becomes:

\[
\mu_d = \left\lfloor \frac{1}{N^2} \sum_{s=0}^{N-1} \sum_{w=s+D-2}^{N-1} |s-w| \right\rfloor = \frac{D}{2}, N = 2m, \tag{5}
\]

\[
\mu_d = \left\lfloor \frac{1}{N^2} \sum_{s=0}^{N-1} \sum_{w=s+D-2}^{N-1} |s-w| = \frac{D(D+1)}{2D+1} \right\rfloor, N = 2m + 1,
\]

where node \( w \) is always within distance \( D \) from node \( s \). The two cases in (6) arise when the same node \( w \) can be reached in \( D \) steps by moving both to the left and to the right from \( s \) (which only happens for the even values of \( N \)). Generalization to higher-order dimensions follows similar guidelines as above and provides an asymptotic result \( \mu_d \approx D/2 \approx dN^{1/d}/4 \) for

\[1\] Alternatively, we can show this result by observing that the binomial distribution in (3) is a sum of \( D \) Bernoulli random variables, which tends to a Gaussian distribution for large \( D \) as given by the Central Limit Theorem.

\[4\] Also note that the size of each dimension (the upper limit of the summations) is given by \( N^{2-d} \).
large $D$ and $N$. Note that the improvement in the average distance in the graph between the circular and non-circular CAN is from $dN^{1/d}/3$ to $dN^{1/d}/4$, or by 25%, while the improvement in the diameter is from $dN^{1/d}$ to $dN^{1/d}/2$, or by 50%.

The distribution of path lengths in CAN is even more interesting. Examine a $d$-dimensional CAN space. The distance between each source node $s = (s_1, ..., s_d)$ and each destination node $w = (w_1, ..., w_d)$ can be decomposed into $d$ random variables:

$$X_i = \|s_i - w_i\|, i = 1, ..., d,$$

where $\|\|$ is a 1D distance norm that depends on whether CAN is circular or not. Since we examine the distribution of all pairs $(s, w)$, each coordinate assumes all possible values and is independent of the other coordinates. Hence, all $X_i$ are i.i.d. random variables with some PMF $p_i(n)$, which is the distribution of $d(x, y)$ in 1D space. Consequently, the probability that a given path from $s$ to $w$ has length $n$ in $d$-dimensional CAN is:

$$p_d(n) = P(X_1 + X_2 + ... + X_d = n).$$

(8)

The density (mass) function of a sum of random variables is the convolution of their individual densities, which means that in order to obtain the PMF of $d(x, y)$, one needs to convolve $p_i(n)$ with itself $d$ times. Thus, we have the following recurrence on the PMF of $d(x, y)$:

$$p_d(n) = p_{d-1}(n) * p_1(n), d \geq 2,$$

(9)

where $*$ denotes discrete convolution. Unlike Chord, CAN does not have a single distribution that describes its shortest paths for all dimensions $d$. To complete the picture, we next derive distribution $p_1(n)$. In non-circular 1D CAN of diameter $D = N - 1$, distribution $p_1(n)$ is given by:

$$p_1(n) = \begin{cases} 
\frac{N}{N^2}, & n = 0 \\
\frac{2(N - n)}{N^2}, & 0 < n \leq N - 1 \\
0, & \text{otherwise}
\end{cases}.$$

(10)

The reasoning behind this equation is simple. For $n = 0$, there are exactly $N$ nodes with zero distances to themselves. For any $n > 0$, there are $N - 2n$ nodes with two neighbors (one on each side), while the remaining $2n$ nodes have only one neighbor. Consequently, the probability $p_1(n)$ for $n > 0$ is given by $[2(N - 2n) + 1 - 2n]/N^2 = 2(N - n)/N^2$. Our simulation results confirm that this formula is exact. It produces skewed distributions with the peak at $D/3$ as shown in Figure 3 (right), which also verifies our prior result in (5).

Circular CAN is very similar, but requires us to distinguish between the even and the odd values of $N$ as shown earlier in (6):

$$p_1(n) = \begin{cases} 
1, & n = 0 \\
\frac{1}{N}, & 2, 0 < n < D \\
1, & n = D \land N = 2m \\
\frac{1}{N}, & 2, n = D \land N = 2m + 1 \\
0, & \text{otherwise}
\end{cases}.$$

(11)

The result in (11) matches simulation results and produces symmetric bell-shaped curves as shown in Figure 4 (left). The figure also provides a Gaussian fit to the 4th-order curve $p_2(n)$. For large $D$, the explanation of the Gaussian shape follows from the Central Limit Theorem as the sum of independent random variables again tends to a Gaussian distribution.

C. Logarithmic CAN

Consider a special case of CAN when its entire routing structure becomes identical to that of Chord. As noted above, when $d = \log_2N/2$, CAN’s degree and diameter are both $\log_2N$, or those of Chord. We call such CAN “logarithmic” and note that the size of its dimensions is $N^{1/d} = 4$ peers. In this case, the 1D distribution of shortest paths given in (11) is reduced to:

$$p_1(n) = \begin{cases} 
1, & n = 0 \\
2, & n = 1 \\
1, & n = 2
\end{cases}.$$

(12)

Now notice that (12) is a binomial distribution (3) with $D = 3$ and $p = 1/2$. Since the sum of $d$ binomial random variables is another binomial variable, all $p_d(n)$ are binomial distributions as well. This result means that not only are logarithmic CAN’s degree-diameter the same as those in Chord, but also that the distribution of its path lengths is identical to that found in Chord.

This is illustrated in Figure 4 (right) for $N = 1,024$, $d = 5$, and $D = 10$, which shows a perfect match between the two graphs (the distributions also match numerically). Even though CAN offers more flexibility with the choice of $d$, suboptimal (non-logarithmic) values of $d$ result in large diameters and are unlikely to be appealing in practical use. From our angle, assuming $d \approx \log_2N/2$, we can conclude that CAN and Chord are essentially the same graph as far as routing is concerned.

D. De Bruijn

From the diameter perspective, de Bruijn graphs offer significantly smaller end-to-end upper bounds on routing time. However, the improvement by a factor of 4 over Chord for $N = 10^6$ no longer holds when we examine the average shortest distances in each graph. Nevertheless, the improvement in
\(\mu_d\) is still substantial, but it drops down to about a factor of 2 faster than Chord as we demonstrate below.

Recall peering rules in de Bruijn graphs. Each node \(v = (v_1, v_2, \ldots, v_D)\) links to all possible neighbors \((v_2, v_3, \ldots, v_D, z)\), \(z \in \Sigma\). Examine a graph of diameter \(D\) and degree \(k\). We next derive how many neighbors of any given vertex \(v\) are expected to be at shortest distance \(n\) from \(v\). Denote by \(S_n = \{u: u = (v_{n+1}, \ldots, v_D, z_1, \ldots, z_n)\} \cup \{z, z_1, z_2, z_3, z \in \Sigma\}\) the set of all neighbors at (not necessarily shortest) distance \(n\) from \(v\), which is produced by shifting vertex \(v\) left \(n\) times and filling the remaining coordinates with arbitrary symbols \(z_1 \ldots z_n\). There are obviously \(k^n\) such vertices. Consider one vertex \(u = (v_{n+1}, \ldots, v_D, z_{n+1}, \ldots, z_1)\) from \(S_n\). Our goal is to find out how many such vertices \(u\) do not belong to \(S_{n-m}\), \(1 \leq m \leq n\) (i.e., there is no path from \(v\) to \(u\) shorter than \(n\) hops).

First consider an arbitrary vertex \(w = (v_{n+1}, \ldots, v_D, y_1, \ldots, y_{n+1})\) from \(S_{n+1}\). Now examine the number of different possibilities that \(u = w\). It is easy to see that among the free variables \(z_1\) and \(y_1\), \(z_1\) is always equal to \(v_{n+1}\), while the remaining variables \(z_2 \ldots z_n\) lead to \(k^{n-1}\) possible pairs \((u, w)\) such that \(u = w\).

However, notice that not all vertices \(v\) allow such matching. The necessary condition on \(v\) is that \(v_n = v_{n+1} = \ldots = v_D\), which leaves \(n-1\) variables \(v_{1} \ldots v_{n-1}\) free and also allows arbitrary choice for \(v_{n}\). Therefore, there are \(k^n\) vertices \(v\) in the graph that allow us to find a match between \(u\) and \(w\). Hence, the expected overlap \(R(n, n-1)\) between \(S_n\) and \(S_{n+1}\) is the probability of \(v\) allowing such overlap times the number of overlapped vertices:

\[
R(n, n-1) = \frac{\binom{k^n}{k^n}}{N}, \quad (13)
\]

Using similar reasoning and subtracting false overlap due to self-loops for all \(m \geq 2\) (self-loops need to be included only once at level \(n-1\) and should be excluded for all other levels), interested reader can show that \(R(n, n-m)\) is

\[
R(n, n-m) = \frac{\binom{k^n}{k^{n-m}}}{N} \left(1 - \frac{1}{k}\right) m, \quad 2 \leq m \leq n, \quad (14)
\]

where \(1-1/k\) is the fraction of self-loops. Next subtract the expected overlap for all levels \(0 \ldots n-1\) from the maximum number of nodes possible at distance \(n\) (i.e., \(k^n\)) and normalize by \(N\):

\[
p(n) = \frac{1}{N} \left( k^n - \sum_{i=0}^{n-1} R(n, i) \right) =
\]

\[
= \frac{1}{N} \left[ k^n - \frac{k^n}{N} \left( 1 - \frac{1}{k} \sum_{i=0}^{n-2} k^i + k^{n-1} \right) \right] =
\]

\[
= \frac{k^n}{N} \left[ 1 - \frac{1}{k} \left( 1 - \frac{1}{k} \sum_{i=0}^{n-2} k^i + k^{n-1} \right) \right] =
\]

\[
= \frac{k^n}{N} \left[ 1 - \frac{k^{n} + k^{n-1}}{kn} \right] =
\]

\[
= \frac{k^n}{N} \left[ 1 - \frac{k^{n} + k^{n-1}}{kn} \right]. \quad (15)
\]

Note that \(n = 0\) is a special case in which term \(k^{n-1}\) should be omitted from the last line of (15). Also notice that the dominating term in the formula is \(k^n\) (especially for large \(N\)). This means that the majority of nodes are located at shortest distance \(D\) from each other and that the average distance is also close to \(D\). Simulations confirm that (15) is exact. It is also easy to verify that \(\Sigma p(n) = 1\) for all values of \(D\) and \(k\).

Next examine the expectation of \(p(n)\):

\[
\mu_d = E[p(n)] = \frac{1}{N} \sum_{n=0}^{D} np(n) \quad (16)
\]

To simplify the expansion, first consider series \(\Sigma x^n\) and notice that it can be computed by differentiating geometric series \(\Sigma x^n\) and multiplying it by \(x\):

\[
\sum_{n=0}^{D} nx^n = \frac{\partial}{\partial x} \left( \sum_{n=0}^{D} x^n \right) = \frac{\partial}{\partial x} \left( \frac{x^{D+1} - 1}{x - 1} \right) \]

\[
= \frac{(D+1)x^{D+1} + x}{(x - 1)^2} = \zeta(x). \quad (17)
\]

Substituting (17) into (16):

\[
\mu_d = \frac{\zeta(1)}{N} - \frac{1}{kN^2} \left[ \zeta(k^2) + \zeta(k) - \zeta(k) \right]. \quad (18)
\]

This generalizes a previously known [4] lower bound on \(\mu_d\) to an exact closed-form solution. For large values of \(k\) and \(N\), (18) simplifies to the following asymptotic shape:

\[
\mu_d \approx D - \frac{1}{k-1}. \quad (19)
\]

As expected, the average distance in de Bruijn graphs cannot shrink much beyond its already very small diameter \(D\). In fact, for large values of \(N\), the approximation in (19) shows that the average distance \(\mu_d\) asymptotically tends to \(D\) as \(k\) becomes large. Nevertheless, \(\mu_d\) in de Bruijn graphs is still half of that in Chord and CAN for the same number of neighbors (assuming non-trivial values of \(N\)). For example, for \(N = 10^6\) and \(k = 20\), \(\mu_d\) in de Bruijn graphs is approximately 4.5, while it is 10 in both Chord and CAN.

E. Discussion

We conducted a study of graph diameters and average distances between any pair of nodes in three classes of graphs. Our results indicate that de Bruijn graphs offer not only provably-optimal diameter \(D\), but also smaller average routing times compared to Chord and CAN. Furthermore, de Bruijn graphs allow the flexibility of selecting node degree \(k\) arbitrarily (i.e., independently of \(N\)) and keeping it constant if so desired. Even assuming a worst-case bound \(\mu_d = D\) for de Bruijn graphs, their average distance log2\(N\) log2\(N\) compares favorably to Chord’s log2\(N\) for all \(N > 16\).

Although there is no available proof regarding the optimality of \(\mu_d\) in de Bruijn graphs, intuitively it is clear that any expander graph that packs nodes so densely must be optimal with respect to the average distance between any pair of nodes. We leave this proof for future work and next investigate clustering and then resilience features of de Bruijn graphs before addressing their practical use in peer-to-peer networks.
V. CLUSTERING

Following significant research effort to model the structure of the current Internet, it was discovered that many of the existing topology generators do not accurately match the "small-world" properties of the Internet graph [3]. These small-world properties are usually described by the clustering coefficient $\gamma$, which determines how tightly neighbors of any node link to each other. In the following, we examine clustering in all three graphs and study graph-theoretic semantics behind the clustering coefficient.

Given graph $G = (V,E)$, node $v \in V$, and its neighborhood $\Gamma = \{u : (v,u) \in E\}$, clustering coefficient $\gamma(v)$ is defined as the ratio of the number of links $L(\Gamma)$ that are entirely contained in $\Gamma$ to the maximum possible number of such links:

$$\gamma(v) = \frac{L(\Gamma)}{|\Gamma|(|\Gamma| - 1)}.$$  \hfill (20)

Graph clustering $\gamma(G)$ is the average of $\gamma(v)$ for all vertices $v$ with degree at least 2. Clustering has been used to explain the structure of social networks when friends of a given person are likely to be acquaintances of each other. However, the main question that we study in this paper is what clustering does it affect the properties desirable in peer-to-peer networks?

A. Clustering Coefficients

In Chord, all nodes are symmetric (modulo $N$) in terms of their connectivity rules. Without loss of generality, consider $v = 0$ and its $k$ neighbors $1, 2, \ldots, 2^k - 1$. Examine any two nodes $x = 2^i$ and $y = 2^j$ ($i < j$) from this list. It is easy to notice that $x$ and $y$ are neighbors if and only if $2^i - 2^j = \Delta$ is a power of two. Hence, we have $\Delta = 2^j (2^i - 1) = 2^i$, for some integer $p$. This can only occur when $2^p - 1 = 1$, or $j-i = 1$. This means that all $k$ neighbors of $v$ are sequentially chain-linked to one another (note that the links are uni-directional from smaller nodes to larger ones). Hence, clustering in Chord is:

$$\gamma(G) = \frac{k - 1}{k(k-1)} = \frac{1}{k}.$$  \hfill (21)

The derivation for CAN is much simpler as one can easily notice that none of the nodes in any neighborhood link to each other. Hence, CAN’s $\gamma$ is zero.

De Bruijn graphs are also fairly simple to analyze. Below, we show that there are exactly $k^2$ nodes with non-zero clustering coefficients, while the remaining $N - k^2$ have $\gamma = 0$. Examine conditions necessary to achieve non-zero clustering for a given node. For any node $v = (v_1, \ldots, v_D)$ and two of its neighbors $a = (a_1, \ldots, a_D)$ and $b = (b_1, \ldots, b_D)$, the necessary condition for non-zero clustering is either $a$ links to $b$ or $b$ links to $a$. Due to symmetric neighboring rules, it is sufficient to analyze the case of $a$ linking to $b$.

First notice that $(a_1, \ldots, a_D) = (v_2, \ldots, v_D, \alpha)$ and $(b_1, \ldots, b_D) = (v_2, \ldots, v_D, \beta)$ for some symbols $\alpha$ and $\beta$ from $\Sigma$. Note that the value of $v_1$ does not affect whether $a$ and $b$ can be neighbors of each other. When there is a directed link from $a$ to $b$, we have for some $z \in \Sigma$: $(a_2, \ldots, a_D-1, z) = b$, or in other words: $(v_1, \ldots, v_D, \alpha, z) = (v_2, \ldots, v_D, \beta)$. This can only occur when $v_2 = v_3 = \ldots = v_D$, which means there are exactly $k^2$ nodes $v$ with this property (since only two out of $D$ coordinates are free variables). If this condition holds, $\alpha$ must equal $v_D$, but $z = \beta$ is another free variable that determines how many pairs $(a,b)$ are neighbors. Since the size of the alphabet is $k$, there are exactly $k$ neighbors of $v$ (i.e., one for each value of $z$) that link to another neighbor. Hence, clustering $\gamma(v)$ is $k/[(k-1)] = 1/(k-1)$.

The average graph clustering $\gamma(G)$ is:

$$\gamma(G) = \frac{1}{N} \left( \frac{k^2 - 1}{k-1} + 0 \cdot (N - k^2) \right) = \frac{k^2}{(k-1)N}. \hfill (22)$$

This holds as long as the diameter of the graph is at least 2 (i.e., $N \geq k^2$). The highest clustering is achieved for $D = 2$ and equals $1/(k-1)$, while for larger diameters it monotonically decays to zero.

B. Cycles

There are two ways to better understand what clustering means and assess its importance for peer-to-peer networks. The first insight is based on cycles. Given a $k$-regular undirected graph $G$, it is easy to notice that the number of 3-cycles per node determines the clustering coefficient of the graph. Recall that an $n$-cycle is a path that starts and ends in the same node and contains exactly $n$ edges. Hence, any 3-cycle must involve two direct neighbors of any node $v$, which results in clustering.

Since in peer-to-peer networks one goal is to reach as many peers as possible within a certain number of hops, cycles that lead back to the original node where the request started are not very helpful. Another goal of peer-to-peer networks is to provide a fault-resilient environment where a simultaneous collapse of several nodes does not separate the graph into disjoint components. Short cycles mean that paths from any node $x$ through different neighbors leading to any destination $y$ must overlap with each other. This is not desirable since when nodes in the neighborhood fail, multiple parallel paths to $y$ can be compromised. This is shown in Figure 5 (left) where failure of node 1 leaves $x$ with only one path leading outside of its neighborhood. On the other hand, if nodes 2 and 3 did not cluster their edges within the neighborhood, $x$ would have four remaining paths to $y$.

Now we come back to the issue of why CAN has zero clustering, but identical shortest-path properties to those found in Chord. The absence of 3-cycles in CAN is explained by the fact that it has no odd cycles whatsoever, but it does have plenty of even cycles. In fact, the number of 4-cycles in CAN is roughly the same as in undirected Chord with the same number of neighbors. Consequently, local properties captured by the clustering coefficient do not necessarily mean much for graphs like CAN where only “friends of friends” have common acquaintances while direct friends of node $x$ never know each other. This is illustrated in Figure 5 (right), where clustering coefficient $\gamma(x)$ is zero, but nodes 2, 3, 4 all link to the same “friend of a friend” node 5.

\footnote{Usually, these paths are required to be node-disjoint, but this always holds for 3-cycles.}
The concept of \( n \)-cycles applies to directed graphs as well; however, it does not directly produce the clustering coefficient due to stricter nature of directed cycles. These difficulties lead us to generalize the framework of clustering using expansion analysis below.

These cases, we seek a more generic and useful definition of though our previous discussion of cycles allows to account for closed-form analytical expression for all three graphs.

Global clustering cycles contribute to graph’s global clustering properties.

- A graph finds “unknown” nodes. Consider graph \( G = (V, E) \) together with set \( S \subseteq V \) and define the set of all edges between \( S \) and \( V \setminus S \) to be \( \partial S = \{ (u,v) : u \in S, v \in V \setminus S \} \). \( \partial S \) is called the edge boundary of \( S \). Edge expansion \( i(S) \) is defined as the ratio of the size of \( \partial S \) to the size of \( S \):

\[
i(S) = \frac{ |\partial S| }{ |S| }.
\]  

It is easy to see the relationship of \( i(S) \) to clustering. Select \( S \) to be the neighborhood \( \Gamma(v) \) of each node \( v \). Therefore, \( |S| = \Delta \) and the number of edges contained within \( S \) is \( k^2 - |\partial S| \), generically assuming a \( k \)-regular graph. Therefore, the clustering coefficient of \( v \) is given by:

\[
\gamma(v) = \frac{k^2 - i(\Gamma)k}{k(k-1)} = \frac{k - i(\Gamma)}{k-1}.
\]  

**Definition.** Graph edge expansion (sometimes called isoperimetric number of the graph) \( i(G) \) is the minimum of \( i(S) \) for all sets \( S \subset V, |S| \leq |V|/2 \)._n

Edge expansion tells us how many edges link outside any set \( S \) (or similarly link inside \( S \)); however, it does not tell us if the outgoing edges link to the same node multiple times. For example, in Figure 5 (left), there are 8 edges leaving neighborhood \( \Gamma(x) \), but they only link to 4 unique nodes, which indicates a good amount of overlap. Edge expansion tells us the size of the cut between \( \Gamma(x) \) and the rest of the graph, which is a useful analysis tool for studying graph’s resilience when edges are expected to fail (i.e., 8 edges in the cut are better than 4). In peer-to-peer systems, node failure is much more common than edge failure, in which case regardless of how many edges cross the cut, the strength of the neighborhood is determined by the number of nodes on the other side of \( \partial S \). Hence, from the resilience perspective of peer-to-peer networks, it makes more sense to examine node expansion of the graph as we define below.

**Definition.** Consider graph \( G = (V, E) \) and set \( S \subseteq V \). Define the node boundary of \( S \) to be \( \partial S = \{ u : (u,v) \in E, u \in S, v \in V \setminus S \} \). Node expansion \( h(G) \) of the graph is given by:

\[
h(G) = \min_{|S| \leq |V|/2} \frac{|\partial S|}{|S|}.
\]  

Notice that \( h(G) \) is always limited above by \( k \). Further notice that the worst condition for choosing set \( S \) is to select nodes that are tightly coupled within set \( S \). This will lead to the minimum outbound expansion as compared to any other set \( S \) of (potentially disjoint or weakly-coupled) vertices. Hence, loosely limiting further derivations to (almost) symmetric \( k \)-regular graphs under study in this paper, set \( |S| \) can be replaced with all possible balls of certain radius \( n \) and the minimum bound in (25) will be preserved.

**A. Chord**

Recall that ball \( B(v,n) \) of radius \( n \) centered at node \( v \) contains all nodes reachable from \( v \) in no more than \( n \) hops. In other words: \( B(v,n) = \{ u : d(v,u) \leq n \} \). It is easy to notice that the boundary of a ball is simply \( \partial B(v,n) = \{ u: d(v,u) = n+1 \} \) and that our derivations in section IV can be applied to study expansion (and global clustering) of each graph.

Both logarithmic CAN and Chord have the same expansion properties since their distributions of \( d(x,y) \) are identical. Hence, from now on, we only consider Chord. The size of each ball of radius \( n \) in Chord is given by (using our prior result in (3)):

\[
|B(v,n)| = N \sum_{k=0}^{n} \binom{k}{2} = \sum_{k=0}^{n} \frac{D!}{(D-k)!}.
\]  

The number of nodes in boundary \( |\partial B(v,n)| \) is \( C_n \) and the resulting expansion of the graph is given by:

\[
h(G) = \min_{|S| \leq |V|/2} \frac{|\partial B(v,n)|}{|S|} = \min_{|S| \leq |V|/2} \left( \sum_{k=0}^{n} \frac{D!}{(D-k)!} \right) \frac{ \left( \frac{D}{n} \right) \backslash n! \right) \frac{n}{C_n}.
\]  

where \( D \) is the diameter of the graph as before. From observing prior plots of the distribution of \( d(x,y) \), it is clear that Chord has a pretty low expansion value \( h(G) \) since the number of nodes in \( \partial B(v,n) \) saturates at \( n = D/2 \). Hence, omitting cases when \( D \) is even and no single ball contains exactly half the nodes, we briefly consider the odd values of \( D \) as they allow us to approach the worst-case lower bound on \( h(G) \). For odd \( D \), \( h(B(v,n)) \) reaches its minimum when ball radius \( n = (D-1)/2 \). Keeping in mind that the size of this ball is \( 2^{D-1} \) and using Stirling’s approximation in (27):
Given destination are expected to be node-disjoint with high probability. Second, path overlap due to little clustering is virtually nonexistent, which means that the maximum possible bound on this metric does not lead to any significant insight either. We should also note that chain-linked de Bruijn graphs are maximum fault resilient as can be shown by combining the results of [41] and [13].

C. Spectral Analysis

What we know so far from classical peer-to-peer network analysis and maximum fault-tolerance metrics is that all three graphs are similar in their resilience. Hence, we seek additional methods that can distinguish between the fault tolerance offered by each graph. Our first step is to analyze algebraic connectivity $\lambda_2$ (the second eigenvalue of the Laplacian spectrum) of each graph [29]. Spectral graph analysis has been recently used model the Internet [19], [44]; however, we take a slightly different approach and focus on the meaning of spectral properties important to peer-to-peer networks.

For a $k$-regular graph $G = (V,E)$, define adjacency matrix $A = \{a_{ij} = 1: i,j \in V, (i,j) \in E; a_{ij} = 0 \text{ otherwise}\}$ and diagonal matrix $F = \{d_{ii} = k; d_{ij} = 0, i \neq j\}$. The Laplacian matrix $L = F - A$ and the Laplacian spectrum of graph $G$ is the set of eigenvalues $\{\lambda_i\}$ of $L$. Sort absolute values $\lambda_i$ of eigenvalues $|\lambda_i|$ in non-decreasing order and recall that the multiplicity of the smallest eigenvalue $\lambda_1 = 0$ is the number of connected components in the graph and that all eigenvalues are between 0 and $2k$ for $k$-regular graphs.

The second largest eigenvalue $\lambda_2$ is called algebraic connectivity and has numerous applications in spectral graph theory. Algebraic connectivity provides upper and lower bounds on various graph metrics, which otherwise would require an NP-hard search through the graph. Generally, larger values of $\lambda_2$ mean faster graph expansion, lower diameter, higher resilience, and better connectivity [10], [29]. For example, trees have very low $\lambda_2 = 1$, which indicates their low tolerance to node failure.

Examine the Laplacian spectrum of all three graphs shown in Figure 6 (left). First, notice that although CAN has a step-function spectrum, its growth is very similar to Chord’s. A similar situation occurred earlier when CAN did not have any odd cycles, but matched Chord’s number of even cycles. Second, notice the difference in algebraic connectivity between CAN/Chord and de Bruijn. In Chord and CAN, $\lambda_2$ is 2, while
in de Bruijn it is 9.4. In general, Chord and CAN keep their \( \lambda_2 \) equal to 2 regardless of the number of nodes \( N \) or degree \( k \), while de Bruijn’s algebraic connectivity is between \( k \) and \( k-1 \). Hence, for all networks with at least 3 neighbors per node, de Bruijn graphs have higher \( \lambda_2 \).

Examine several useful results related to graph’s algebraic connectivity. Recall that bisection width \( b(G) \) is the minimum number of edges whose removal splits the graph into two equal-size disjoint components (this problem is \( NP \)-complete). Bisection widths indicate how difficult it is to separate the graph into two giant (i.e., order of \( N \)) components by failing individual edges. The following provides a lower bound on \( b(G) \) [29]:

\[
b(G) \geq \frac{N}{4} \lambda_2(G) + o(1). \tag{30}
\]

Thus, the lower bound on bisection width is \( N/2 \) for Chord and logarithmic CAN and no less than \( N(k-1)/4 \) for de Bruijn. In fact, the \( N/2 \) bound is tight for logarithmic CAN. To partition CAN into two disjoint halves, one needs to create two \((d-1)\)-dimensional planes that separate the graph into two partitions as shown in Figure 6 (right) for \( d = 2 \). Since the size of each dimension in logarithmic CAN is 4 (i.e., \( N^{1/d} \)), where \( d = \log_2(N)/2 \), the size of each \((d-1)\)-dimensional plane is \( N/4 \). One needs two such planes to cut through the graph to completely separate the nodes, which results in \( N/2 \) edges in the cut. Note that non-circular CAN needs only one plane, or \( N/4 \) edges to become partitioned.

Given the same de Bruijn graph in Figure 6 (left), we have \( b(G) \geq 2.35N \), which is almost 5 times higher than a tight (achievable) bound in CAN. Although (30) is just a lower bound, it provides some indication about the strength of the bound in CAN. To partition CAN into two disjoint halves, one needs to create two \((d-1)\)-dimensional planes that separate the graph into two partitions as shown in Figure 6 (right) for \( d = 2 \). Since the size of each dimension in logarithmic CAN is 4 (i.e., \( N^{1/d} \)), where \( d = \log_2(N)/2 \), the size of each \((d-1)\)-dimensional plane is \( N/4 \). One needs two such planes to cut through the graph to completely separate the nodes, which results in \( N/2 \) edges in the cut. Note that non-circular CAN needs only one plane, or \( N/4 \) edges to become partitioned.

As noted before, larger edge expansion \( i(G) \) is desirable since it keeps any neighborhood \( S \) connected to the rest of the graph with at least \( i(G)|S| \) edges. Also notice that the lower bound on \( i(G) \) is again given by \( \lambda_2 \), which leads to another way of showing that de Bruijn graphs are tighter connected graphs than CAN/Chord.

Spectral analysis above provides several ways of understanding the differences between the classical approaches (i.e., Chord/CAN) and de Bruijn graphs not previously available in peer-to-peer literature. Larger values of algebraic connectivity in de Bruijn point towards higher resilience against both edge and node removal in addition to their optimal routing established earlier. Below, we examine several other resilience metrics, which may lead to even better understanding of de Bruijn’s very high fault tolerance.

VIII. PATH OVERLAP AND SIMULATIONS

So far, we have been omitting a great deal of simulation results since most of the derived formulas in this paper were exact. In this section, we examine several empirical metrics and back our analysis of these metrics with simulations.

Define set \( P(x,y) \) to contain all vertices along some path from \( x \) to \( y \). Denote by \( Q(x,y) \) the set of all vertices in \( P(x,y) \) except \( x \) and \( y \): \( Q(x,y) = P(x,y) \setminus \{x,y\} \). Graph’s \( k \)-node-connectivity means that for every pair of nodes \( (x,y) \), there are exactly \( k \) pair-wise non-overlapping paths \( P_i(x,y), ..., P_k(x,y) \): \( Q(x,y) \cap Q(x,y) = \emptyset \) for all \( i \neq j \). Node-disjoint paths are very attractive to peer-to-peer networks as they provide independent backup routing options when the main shortest path fails.

Now notice that even though all three graphs under study have \( k \) node-disjoint paths for each pair \( (x,y) \), not all of these paths can be found using greedy routing at each node. In fact, in order to deterministically find all \( k \) non-overlapping paths between \( x \) and \( y \) in a generic graph, one needs to flood the entire graph using breadth-first search or similar techniques. Below we examine how well the routing rules in each of the graph are able to find non-overlapping paths and what happens to the diameter of the graph when nodes along the best path are failed.

For any pair of nodes \( (x,y) \), define \( P_i(x,y) \) to be the shortest (according to the greedy routing rules of the corresponding graph) path to \( y \) through \( x \’s \) neighbor \( i \). The reason for using shortest paths is because when the best neighbor towards \( y \) fails, the graph routes through the second-best neighbor also trying to achieve the shortest path to \( y \). Hence, if these two paths overlap, the diameter or connectivity (or both) may deteriorate. Define \( T(x,y) \) to be the total number of vertices in all shortest paths \( P_i(x,y) \) from \( x \) to \( y \) and \( U(x,y) \) to be the number of unique vertices in all such paths: \( T(x,y) = \sum P_i(x,y) \) and \( U(x,y) = |\cup P_i(x,y)| \). Further define the average percentage \( U \) of unique nodes in all parallel paths:

\[
U = \frac{\sum_x \sum_y U(x,y)}{\sum_x \sum_y T(x,y)} . \tag{32}
\]

Finally define path overlap \( J \) to be \( 1-U \). We demonstrate the performance of the graphs using the same example of \( N=11 \):
1,024 for Chord and \( N = 1,000, k=10 \) for de Bruijn. Path overlap \( J \) is 37% in Chord and only 3% in de Bruijn. The difference between Chord and de Bruijn is significant, but not quite unexpected given our previous discussion of expansion properties of each graph. This result means that de Bruijn (automatically) selects backup paths that do not overlap with the best shortest path or each other.

Next examine Figure 7, which shows the number of non-overlapping paths between any pair of nodes in the corresponding graph (we exclude pairs \((x,x)\) and direct neighbors from the figure). Interestingly, 50% of pairs \((x,y)\) in Chord have a single shortest path \( P_1(x,y) \), which overlaps with every other shortest path \( P_j(x,y) \). Hence, when nodes along this path fail, many alternative paths are likely to be affected. The right side of the figure shows that de Bruijn graphs virtually always have exactly 8 non-overlapping paths between any pair of nodes. This means that when nodes fail and packets get re-routed along the optimal paths of each neighbor, they have very little likelihood of encountering the already-failed nodes. Qualitatively, this difference leads to better fault-resilience of de Bruijn graphs and smaller diameter under node failure.

![Figure 7](image_url)

Figure 7. Distribution of the number of non-overlapping shortest paths in Chord (left) for \( N = 1,024 \). The same distribution in de Bruijn (right) for \( N = 1,000, k = 10 \).

In our next experiment, we introduced adversarial failures into the network. We failed all nodes along the shortest path from \( x \) to \( y \) and routed traffic through the second-best neighbor (i.e., the neighbor that is expected to have the shortest distance to \( y \) among the remaining neighbors). Then we failed all nodes along the second-best path and examined the third-best path, and so on. The distribution of average path lengths in the graph is shown in Figure 8. The average distance in Chord rises to as high as 19 hops when routed through some of the “suboptimal” neighbors. De Bruijn graphs maintain the same low diameter and the average distance rises by 1 for all but the very last neighbor.

![Figure 8](image_url)

Figure 8. Distribution of shortest-path distances in Chord (left) and de Bruijn (right) under adversarial failures.

Finally, we revisit the common node failure methodology of previous work – fail \( p \) percent of all nodes randomly and observe the structure of the resulting graph. Even though the graphs are highly resilient to node failure and the giant connected component often contains over 95% of alive nodes, not all graphs are able to find the necessary paths using greedy routing. Define \( f(p) \) to be the fraction of all alive pairs \((x,y)\) that cannot be routed using greedy rules of the corresponding graph under \( p \)-percent failure. Table 3 shows that Chord’s fraction \( f(p) \) quickly rises and reaches over 20% for \( p = 0.5 \). These 20% of the cases occur when greedy routing rules lead to infinite cycles and Chord has no way of reaching the destination except flooding. De Bruijn graphs demonstrate a much smaller susceptibility to routing problems as also shown in the table.

<table>
<thead>
<tr>
<th>( p )</th>
<th>Chord ( f(p) )</th>
<th>Diameter</th>
<th>Average distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.5%</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>0.1</td>
<td>1.9%</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>0.2</td>
<td>5.3%</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>0.3</td>
<td>7.2%</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>0.5</td>
<td>21%</td>
<td>21.6%</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 3. Greedy routing performance under \( p \)-percent failure for Chord \((N = 1,024)\) and de Bruijn \((N = 1,000, k = 10)\).

IX. OPTIMAL DIAMETER AND RESILIENCE INFRASTRUCTURE

We have accumulated sufficient evidence that shows that de Bruijn graphs possess both short routing distances and high fault tolerance. In this section, we discuss ODRI — Optimal Diameter and Resilience Infrastructure, which builds de Bruijn graphs incrementally and preserves their nice properties at the application layer. Luckily, de Bruijn graphs are very simple to build incrementally and most of the details (some of which we skip) are almost identical to those in Chord.

Let \( N_{max} \) be the maximum possible number of nodes in the system. Organize the space of all possible nodes between \([0, N_{max} -1]\) into a modulo-\( N_{max} \) number field. Now notice that each node \( x \) in de Bruijn graphs is a base-\( k \) integer \( N(x) \) and that its neighboring rules can be expressed as:

\[
N(x) \rightarrow (N(x)k + i) \mod N_{max}, \quad i = 0, \ldots, k - 1,
\]

since a shift left by one digit is equivalent to multiplication of \( N(x) \) by \( k \). The rest of ODRI is very similar to Chord. Each node holds a consecutive stretch of the number space, which can be denoted by \([z_1, z_2]\), for some \( z_1, z_2 \). A node joins and routes to the area of the circle where its hash index \( N(x) \) is located. Previous owner \( y \) of zone \( Z = [z_1, z_2] \) where \( N(x) \) falls splits the zone and gives \( x \) a continuous part of \( Z \) that contains \( N(x) \) as one of its boundaries and keeps the remainder where its own hash index is located.

Consider node \( x \) that owns zone \([z_1, z_2]\). Each of the integer values in \([z_1, z_2]\) corresponds to the underlying de Bruijn graph of size \( N_{max} \). Hence, to preserve de Bruijn linkage at the application layer, \( x \) must link to all peers holding the other end of each edge originating in \([z_1, z_2]\). This means that there is an application-layer edge \((u,v)\) if and only if there is an edge \((u,v)\) in the underlying de Bruijn graph such that \( u \in \mathbb{Z}_k \),
and $v \in Z_x$, where $Z_x$ and $Z_y$ are the corresponding zones held by $x$ and $y$.

Theorem 1. ODRI maintains application-layer degree equal to $k$ with high probability (i.e., under uniform user joins).\footnote{This is a standard assumption that stems from randomness of the hashing function and independence of hash indexes of any two nodes [42].}

Proof. Uniform joins mean that each node holds an equal-size zone. Denote by $M$ the size of each zone and consider an arbitrary node $x$ with zone $[z, z+M-1]$. Notice that $x$ links to every peer whose zone falls between $zk$ and $(z+M-1)k+1$. Since this stretch spans $Mk-1$ de Bruijn vertices from the underlying graph, there are exactly $k$ different peers in this stretch. Using similar reasoning, it is easy to show that the indegree of each node is also $k$.

Notice that building the routing table for a newly joined node requires only $O(1)$ message complexity as it can be copied from the previous owner of the zone (this is not the case in other protocols, which often require $O(\log N)$ messages). Furthermore, given the result of theorem 1, it is easy to show that the application-layer graph in ODRI is a scaled-down version of the underlying de Bruijn graph (where $N_{\text{max}}$, and each coordinate are scaled by $M$). For example, every vertex $z$ in the underlying graph links to $k$ vertices $(zk+i) \mod N_{\text{max}}$, while every peer $x = z/M$ links to $k$ peers $(zk+i) \mod N$ as shown in the proof of theorem 1. Thus we have the following theorem to conclude the paper.

Theorem 2. ODRI maintains an $N$-node application-layer de Bruijn graph with diameter $\left\lceil \log_{k} N \right\rceil$ with high probability (i.e., under uniform user joins).

This result shows optimality of the peer-to-peer application-layer graph and also means that all results derived earlier in this paper can be directly applied to ODRI’s peer-to-peer graphs.

X. CONCLUSION

At this stage of peer-to-peer research with an overwhelming number of recent proposals, it is hard to assess the benefits of one peer-to-peer network over another without a unifying analytical framework that can capture graph-theoretic properties of each proposal.

This paper conclusively answered the diameter-degree tradeoff question of current DHT peer-to-peer research and conducted an extensive graph-theoretic comparison of proposed methods in terms of their routing and fault resilience. The architecture proposed in this work relies on de Bruijn graphs that achieve optimal diameter for a given fixed degree, offer optimal resilience ($k$-node connectivity), and provide almost maximum node expansion that guarantees very little overlap between parallel paths to any destination. Furthermore, de Bruijn graphs possess much higher bisection width and resilience to edge removal than the existing methods. Combining all these findings together with incremental construction of ODRI, we conclude that de Bruijn graphs are viable and appealing structures for peer-to-peer networks.

REFERENCES


